(2)

UNCLASSIFIED

AD NUMBER ____________________

TECOM PROJECT NO. 7-CO-R89-EPO-007

METHODOLOGY INVESTIGATION FINAL REPORT OF SOFTWARE MATURITY MODEL VALIDATION

BY

KEN VAN KARSEN

Software and Interoperability Division
Electronic Technology Test Directorate
U.S. ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA 85613-7110

13 NOVEMBER 1989

Prepared for:
U.S. Army Test and Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

UNCLASSIFIED

DEPARTMENT OF THE ARMY
HEADQUARTERS, U.S. ARMY TEST AND EVALUATION COMMAND
ABERDEEN PROVING GROUND, MARYLAND 21005 – 5055

REPLY TO
ATTENTION OF

AMSTE-TC-D (70-10p)

23 AUG 1990

MEMORANDUM FOR Commander, U.S. Army Electronic Proving Ground, ATTN: STEEP-CT-E (Mr. K. Karsen), Fort Huachuca, AZ 85613-7110

SUBJECT: Methodology Investigation Final Report of Software Maturity Model Validation, TECOM Project No. 7-CO-R89-EPO-007

1. Subject report is approved.

2. Point of contact at this headquarters is Mr. Richard V. Haire, AMSTE-TC-D, amstetcd@apg-emh4.apg.army.mil, AV 298-3677/2170.

FOR THE COMMANDER:

[illegible]
FREDERICK D. MABANTA
Chief, Tech Dev Div
Directorate for Technology

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION US Army Electronic Proving Ground 6b. OFFICE SYMBOL (If applicable) STEEP-ET-S	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Fort Huachuca, Arizona 85613-7110	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION US Army Test & Eval Cmd 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Aberdeen Proving Ground, MD 21005	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
Methodology Investigation of Software Maturity Model Validation

12. PERSONAL AUTHOR(S)
Ken Van Karsen

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM ________ TO ________	1990/11/13	78

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software Test and Software Reliability

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report covers phase I of an investigation to identify methods of quantitatively assessing software reliability. A set of candidate software reliability models were screened for potential use in estimating reliability parameters. Other techniques were examined briefly. Finally, the various methods were evaluated with respect to applicability during developmental testing. (KR)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mr. Curtis Massie	22b. TELEPHONE (Include Area Code) (602) 533-8204 22c. OFFICE SYMBOL STEEP-ET-S

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

SECTION 3. APPENDIXES

LIST OF TABLES

LIST OF FIGURES

FOREWORD

CONTENTS

This report on the methodology investigation of the software maturity model validation represents the completion of Phase I of the investigation. In Phase II, the models recommended by this report will be applied to a tactical system for validation.

This report has been developed in accordance with Test and Evaluation Command (TECOM) Reg 70-12 and consists of the following sections and appendixes:

a. Section 1 is an executive summary of the investigation.

b. Section 2 contains the details of the investigation.

c. Section 3 consists of the following appendixes:

Appendix A - Methodology Investigation Proposal

Appendix B - References

Appendix C - Acronyms and Abbreviations

Appendix D - SMERFS Models

Appendix E - Glossary

Appendix F - Distribution

ACKNOWLEDGMENTS

The following personnel from Comarco, Inc. assisted in the preparation of this report under Contract Number DAEA18-87-C-0014:

Mr. Britt Barrett compiled this final report.

Mr. Fred Gampper provided the technical direction for this report.

Ms. Sharon Vanderhyden and Ms. Karen Norris provided skillful assistance in the technical editing and word processing of the report.

The following personnel from USAEPG assisted in the preparation of this report under Contract Number DAEA18-87-C-0014:

Ms. Linda Skjerven and Ms. Cassondra Renfro provided helpful comments in the review of the report.

SECTION 1. SUMMARY

1.1 BACKGROUND.

Software has become a major part of Command, Control, Communications, and Intelligence (C^3I) systems. The complexity of software system development and maintenance is steadily increasing. Test data shows that software errors occur more often than hardware errors, and that many software errors are undetected until the system is tested in the field. The cost to correct software errors increases as detection is prolonged through the software life cycle. These are some of the reasons why software reliability is becoming increasingly important (reference 1). Software reliability is probably the most important factor of software quality (reference 2). It has become, in fact, vital for software managers and engineers to be able to measure and predict software reliability before fielding of systems.

System reliability is measured in stochastic terms. That is, system reliability is "the probability that the system performs its assigned functions under specified environmental conditions for a given period of time." The use of reliability as a rating factor for a system is intimately associated with the need for the system to function properly, over a specified period of time, when such operation is of a critical nature. According to one Air Force study, "In the past, the approach to determining or predicting system reliability has been to look at the hardware components, calculate their combined reliability, assume software reliability was one, and use the hardware reliability number as the system reliability" (reference3). That study exposes the inadequacy of this approach. It indicates that "software is a significant contributor to system failures" and it identifies software reliability models as one dimension of research to improve system reliability prediction and estimation.

Software reliability may be characterized in terms that closely parallel the definition of reliability for technical systems. Goodenough defines software reliability as "the frequency and criticality of program failure where failure is an unacceptable effect or behavior under permissible operating conditions." Like hardware, software reliability can be represented by the rate at which errors are uncovered and corrected. Unlike hardware, there is less evidence that empirical error data (collected during testing and after release of the software) can be used to develop accurate predictive models of software reliability.

It is difficult to give a precise definition of software reliability. Many attempts have been made to standardize the definition; however, no one definition is accepted as standard (reference 1). Some might say that software is reliable if it is correct. That is, software is reliable if it meets its initial specifications and performs as specified. This definition does not take into account the possibility that the software specifications may be incomplete and incorrect. This definition confuses software reliability with software correctness. Software reliability concerns any software failure, whereas software correctness concerns the degree to which software design and code conform to specifications and standards (reference 4).

Musa defines software reliability as "the probability of failure-free operation of a computer program for a specified time in a specified environment" (reference 2). This methodology report chooses this definition because the concern of this investigation is to ascertain software reliability measures which can be combined with hardware reliability measures to determine system reliability. The definition is tied to the idea that the reliability of a software system (i.e., hardware, software, and manual operations) is dependent upon the reliability of its hardware and software components. In making this choice, this report recognizes the dependence of software reliability on other software quality factors such as correctness and maintainability. This interdependence of software quality factors is the subject of another methodology investigation report (reference 5).

In Musa's definition of software reliability, failure-free is defined as having no occurrence of a software failure. Software failure is defined as a deviation of the operation of a computer program from its requirements (reference 2). Software failure and software error are used interchangeably in the literature, and this is the source of much confusion. This is because software fault and software error are also used interchangeably. Software error and software fault, therefore, are often equated. A software fault, however, is not the same thing as a software failure. The problem is resolved by realizing that software error has two meanings. In one context, software error means software failure. In another context, software error means software fault. To avoid confusion, this report does not equate software fault and software error. It does, however, use software failure and software error interchangeably because many of the cited sources equate these two terms.

Musa's definition of software failure implies that software failure is a dynamic process. That is, the program has to be executing for a failure to occur. Software failures can be characterized in the following ways: time to failure, time interval between failures, cumulative failures experienced up to a given time, and failures experienced in a time interval. Since much of the literature uses software error and software failure interchangeably, software failures are also characterized as time between error, cumulative errors experienced up to a given time, and errors experienced in a time interval.

Software faults cause software failures. A software fault is a defect introduced into software through human error. A software fault is created when a programmer makes a coding error. Faults are also created when a systems analyst incorrectly specifies a requirement or when a programmer analyst produces erroneous program design language (PDL). Each of these latter instances can lead to seemingly correct code which, when executed, propagates an erroneous requirement or design.

Program size and complexity have grown to the point where it is impossible to check the astronomical number of logic paths through the code (reference 6). For example, large scale real-time embedded systems such as the Trident-I Fire Control System (TFCS) have an astronomical number of logic paths (reference 7). It is impossible to check every conceivable logic path in its computer code. Software reliability estimation is an area of research which attempts to quantify the number of faults remaining in a program without having to check out all of these paths. More importantly, this form of estimation can tell us how often the faults cause failures. As for hardware, this is the primary objective of software reliability prediction: given a component (software component), what is the probability that it will fail in a given time period, or equivalently, what is the expected time duration between failures? In hardware, if the mean time between failure (MTBF) is too small, then more reliable components or redundant components are used to achieve the required improvements. In software, the approach to improved reliability is replacing erroneous code with debugged code.

Over the past two decades, many models and estimation procedures have been proposed to quantify the reliability of software. Examples of such models are mathematical models known as software reliability models. Dr. William H. Farr of the Naval Surface Warfare Center conducted a survey of reliability modeling and estimation techniques in which he identified three categories of software reliability models (reference 7): error seeding models, data domain models, and time domain models. Dr. Amrit Goel of Syracuse University categorized software reliability models in a similar way through a survey of his own (reference 3).

Error seeding/tagging models involve "seeding" software with a known number of software faults. These models assume that the actual distribution of software faults is the same as the distribution of the "seeded" faults. The total number of software faults inherent in the software are estimated from counts of software faults discovered during software testing.

Data domain models estimate a program's current reliability based on the ratio of the number of successful runs observed to the total number of runs made. That is, the estimated reliability is simply the total number of successful runs divided by the total number of test runs. Run is an arbitrary term generally associated with some function software performs (reference 2).

Time domain models have received the greatest emphasis in the literature and in real world applications. These models find their roots in hardware reliability modeling. That is, the concepts of hardware reliability modeling were adapted for use in modeling software reliability. Some of these models, however, have terms which do not have hardware counterparts (e.g., the number of remaining faults). Each of these models make assumptions that can vary from model to model (reference 7). One of the assumptions that the Geometric Poisson Model makes, for example, is that each software fault that is discovered is either corrected or not counted again. Brooks and Motley's Models, on the other hand, assume that software faults can be reintroduced in the software fault correction process. As another example, Musa's Execution Time Model assumes that the software failure rate is constant and changes only at each software fault correction. Moranda's Geometric Model, however, assumes that the software failure rate is initially a constant which decreases in a geometric progression as software failures are detected. Each of these models makes certain independence assumptions. For example, Moranda's Geometric Model assumes that the detections of software failures are independent. This assumption is needed to model software failure as an exponentially decaying process. The independence assumptions all of these models make are often challenged. There is, however, considerable evidence that the assumptions are valid (reference 2). The confusion arises from the argument that software faults can be related, and hence are not independent. The possibility of related faults, however, does not imply that software failures are related because software failures occur as the result of randomization of inputs.

One category neither survey addresses are software reliability models based on the internal characteristics of the program. These models provide a priori estimates of software failures. That is, they predict the number of software failures before operational data is available. Dozens of such models estimate the number of faults in a program based on static characteristics of the program itself which are generally related to software complexity measures (reference 8). These models predict the total number of failures using the fault reduction ratio which is the number of inherent faults in the program divided by the fault-reduction factor. The fault reduction factor is estimated from empirical data involving previous software development projects. There is evidence that this factor is project independent, although such a value is not known at present (reference 2).

People who make these models do not propose they be used without checking the reasonableness of their assumptions. The current trend is to incorporate one or more time domain models in a software package which includes routines to perform statistical analysis of the models. These packages include options to check the reasonableness of the model assumptions by seeing how well the model fits the data (references 6).

The Department of Defense (DoD) has issued a directive which addresses the reliability problem (reference 9). Department of Defense Directive (DoDD) 5000.3, "Test and Evaluation," authorizes the issuance and publication of DoD 5000.3-M-1, "Test and Evaluation Master Plan (TEMP) Guidelines." According to these guidelines (reference 10), the TEMP is "the basic planning document for all test and evaluation (T&E) related to a particular system acquisition." This document states, "The TEMP shall contain criteria usable for assessment of software maturity." It indicates that evaluation criteria should include quantitative thresholds for the Initial Operational Capability (IOC) system at Milestone I, Milestone II, and for the mature system.

The TEMP Guidelines distinguish between maturity and reliability. Maturity subsumes reliability. This becomes clear if one thinks of reliability as dependability. Software can be dependable (not fail), yet it can still be poorly documented (not be maintainable). According to the TEMP Guidelines, in order for a system to be mature, it "must have achieved its reliability thresholds and be fully maintained in accordance with the DoD Component's maintenance concept."

The TEMP Guidelines define reliability to be "the probability that an item will perform its intended function for a specified interval under stated conditions." Threshold is defined to be "a minimum level of performance required at a point in a system's life cycle such that the threshold at maturity equals the requirement." In view of this, a system must have reliability thresholds that are quantitative and probabilistic. Since the system includes software, there should also be software reliability thresholds. To measure such thresholds, techniques which are quantitative and probabilistic are required. The area of software reliability estimation arose for this purpose.

1.2 PROBLEM.

DoDD 5000.3-M-1 requires quantitative, probabilistic estimates of software reliability to help ascertain software maturity. Although software reliability models which compute such estimates exist, their suitability with respect to the availability of required data has not been determined. Furthermore, none of these models have been validated with test data from the field for test items requiring evaluation by the United States Army Test and Evaluation Command (TECOM).

1.3 OBJECTIVE.

The objective of this investigation is to establish an accepted method of computing software reliability to help assess the maturity of software in embedded computer resources (ECR). Specific goals are:

a. To develop, as part of Phase I, a set of initial candidate software reliability models to be screened for potential use in estimating software reliability.

b. To identify, as part of Phase I, approaches other than reliability models to estimate software reliability.

c. To complete, as part of Phase I, an evaluation of the set of candidate software reliability models and other approaches to see if they can be applied to developmental testing (DT).

d. To provide, as part of Phase II, recommendations for a set of models or any other methods to help determine the status of software during DT.

1.4 PROCEDURES.

Software reliability models and other methodologies for assessing software maturity which are currently available from industry, academia, and government agencies were identified through a survey in one or more of the following ways: attendance at appropriate seminars, examination of the literature, and consultation with other organizations.

Once identified, the models and methods were evaluated based on the following criteria:

a. Does the model or other method provide probabilistic and quantitative estimates of program reliability?

b. Is the model or other method sufficiently documented?

c. Does the model or other method have realistic data requirements?

d. Is the model or other method readily available?

e. Is the model or method applicable to DT?

1.5 RESULTS.

The survey resulted in the identification of numerous software reliability models (Table 1.5-I). Most of these models were previously identified by the Naval Surface Warfare Center (NSWC) in a comprehensive survey of their own (reference 7).

One of the major findings of the investigation was the identification of the Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) package. SMERFS was developed by the NSWC to help assess the software maturity of large scale real-time systems such as Trident II. It is an interactive program for measuring and predicting software reliability. The models chosen for SMERFS are a subset of those seen in Table 1.5-I plus one additional model that is an adaptation of one of these models. The SMERFS models, seen in Table 1.5-II, were chosen "for their performance in comparative studies and their ability to handle data collected from various testing environments" (reference 6).

The survey of the current investigation resulted in the identification of three alternate approaches for measuring software maturity. One of these approaches is attributable to the TECOM Army Materiel Test and Evaluation Directorate at White Sands Missile Range (WSMR), New Mexico (reference 11). Two other approaches include methods outlined in Air Force Operational Test and Evaluation Center Pamphlet (AFOTECP) 800-2, Volume 1, "Software Operational Test and Evaluation Guidelines" (reference 12), and in Army Materiel Command Pamphlet (AMC-P) 70-14, "Army Materiel Command Software Quality Indicators" (reference 13).

The evaluation of models and other methods resulted in the selection of a set of software reliability models for potential use in estimating software reliability. Only models and other methods satisfying the evaluation criteria were selected as candidates for DT. Based on these criteria, only the SMERFS were found to be of potential use for DT. The Software Performance Parameter Assessment (SPPA) model (reference 14), although statistically sound, was ruled out as a candidate, for example, because it is poorly documented and has unrealistic data requirements. The approach developed by WSMR was eliminated as a candidate, for example, because it is not fully documented, and it does not provide probabilistic, quantitative estimates of software maturity. It only gives subjective assessments of software maturity. Furthermore, it does not address software reliability. The approach indicated in AMC-P 70-14 was also eliminated because it does not apply to DT, nor does it provide quantitative, probabilistic estimates.

The evaluation of models and other methods resulted in the realization that the Test Incident Report (TIR) format does not provide for the collection of all data needed to use these models or methods. For example, the SMERFS model data requirements are exhibited in Table 1.5-III. TIRs do not provide for the starting time of testing, the ending time of testing, and the Central Processing Unit (CPU) time expended between software error occurrences. TIRs do provide the wall clock time of error occurrences and the chargeability of such occurrences (e.g, software).

Table 1.5-I. Software Reliability Models

ERROR SEEDING/TAGGING MODELS

. Mills Seeding Model
. Rudner Seeding Model
. Basin Tagging Model

DATA DOMAIN MODELS

. Nelson Model
. LaPadula's Reliability Growth Model

TIME DOMAIN APPROACH MODELS

Classical Software Models

. Weibull Model
. Shooman Model
. Jelinski – Moranda De – Eutrophication Model
. Schick – Wolverton Model
. Generalized Poisson Model
. Geometric Model
. Schneidewind's Model
. Non – Homogeneous Poisson Process
. Duane's Model
. Musa's Execution Time Model
. Brooks and Motley's Models

Bayesian Models

. Littlewood's Debugging Model
. Littlewood and Verrall's Reliability Growth Model
. Thompson and Chelson's Reliability Growth Model

Markov Models

. Software Performance Parameter Assessment
. Trivedi and Shooman's Many State
. Littlewood's Semi – Markov Model

1.6 ANALYSIS.

During the investigation, some confusion was found regarding the terms "software reliability" and "software maturity." Software reliability should not be confused with software maturity. Nor should the two terms be considered divorced from one another. Software maturity encompasses software reliability. Software maturity is defined as "a measure of the software's evolution toward satisfying all documented user requirements (reference 12)."

Table 1.5-II. SMERFS Reliability Models

WALL CLOCK OR CPU TIME MODELS

1 The Littlewood and Verrall Bayesian Model
2 The Musa Execution Time Model
3 The Geometric Model
4 The NHPP Model for Time Between Error Occurrence

ERROR COUNT MODELS

1 The Generalized Poisson Model
2 The Non – Homogeneous Poisson Model
3 The Brooks and Motley Model
4 The Schneidewind Model

This includes reliability requirements. When one measures software maturity, one is measuring all the factors making up software. A previous methodology investigation addressed the problem of measuring all software factors (reference 5). The current investigation, however, focused on measuring one aspect of software maturity, namely, software reliability.

Although SMERFS was selected for potential application to DT, other reliability models could do as well. SMERFS is simply a representative set of reliability estimation models which satisfies all the selection criteria and happens to be readily available.

Of all the tools identified, SMERFS holds the most promise for several reasons. First, it produces probabilistic and quantitative estimates of software reliability, which is what the TEMP Guidelines require. Other approaches do not provide precise estimates of software reliability. Furthermore, SMERFS is well documented; other approaches however, are not documented at all. Also, SMERFS has been applied with varying degrees of success by International Business Machines (IBM) on the Space Shuttle Program (reference 15), the United States Navy on the Trident Missile Program (reference 16), and Hughes Aircraft on an air defense system (reference 17).

Table 1.5-III. SMERFS Model Data Requirements

MODEL	MODEL DATA REQUIREMENTS
Error Count Models 1 The Generalized Poisson Model 2 The Non – Homogeneous Poisson Model 3 The Brooks and Motley Model 4 The Schneidewind Model	. The number of errors detected in each interval of testing (applies to models 1, 2, 3, and 4). . The lengths of the various testing intervals (applies to models 1, 2, and 3). . The number of errors corrected at the end of each testing period (applies to model 1). . The total number of testing periods (applies to model 4).
Wall Clock Time Models 1 The Littlewood and Verrall Bayesian Model 2 The Geometric Model 3 The NHPP Model for Time Between Error Occurrence	. The wall clock starting and ending times of each interval of testing (applies to models 1, 2, and 3). . The wall clock time of occurence of each error during each interval of testing (applies to models 1, 2, and 3).
CPU Time Models 1 The Littlewood and Verrall Bayesian Model 2 The Musa Execution Time Model 3 The Geometric Model 4 The NHPP Model for Time Between Error Occurrence	. CPU time expended between successive occurrences and/or non – occurrences of errors (applies to models 1, 2, 3, and 4). . Testing compression factor (applies to model 2).

Finally, SMERFS is available for use on microprocessor systems and it is free of charge.

Although the data requirements of SMERFS are realistic, a problem remains with data collection. Based on data from TIRs alone, none of the SMERFS models can be run. For example, since TIRs do not provide CPU time expended between software errors, the CPU option cannot be chosen for any of the time between error models. As another example, since TIRs do not provide the starting and ending times of testing, none of the error count models can be run. For this same reason, the wall clock option cannot be chosen for any of the time between error models.

1.7 CONCLUSIONS.

The Phase I goals of the software maturity model investigation were achieved. The following conclusions were drawn:

a. Models for estimating software reliability abound. The survey of software reliability models was accomplished. A set of candidate software reliability models for potential use in estimating software reliability during DT was developed.

b. Other approaches for assessing software reliability were identified.

c. An evaluation of these models and other approaches was completed, providing a set of models for potential application to DT. Of the methods evaluated, software reliability models were found to be better suited for making reliability estimations. TIRs, however, do not currently provide enough fields of information to allow for the collection of data needed to run these models. Until TIRs are upgraded to include such information, the data needed to run these models will have to come from elsewhere.

d. The overall objective of establishing an accepted method of computing software reliability to help assess software maturity depends upon the successful outcome of Phase II.

1.8 RECOMMENDATIONS.

The following recommendations are suggested as a result of this investigation:

a. To complete the investigation, it is recommended that Phase II begin so recommendations for a set of models to help determine software maturity during DT can be provided. The set of candidate software reliability models chosen during Phase I should be demonstrated by applying data from a selected tactical system, and the test results should be evaluated.

b. It is recommended that Army TIRs be enhanced to include fields for the following data: CPU time expended between software error occurrences; starting and ending times of testing.

c. If the Phase II demonstration proves successful, then the methodology of using reliability models such as those in SMERFS should be documented in a Test Operations Procedure (TOP). For example, the TECOM TOP for Software

Testing, dated 15 November 1977, could be revised to address software reliability as a factor of software maturity. The methodology provided in this report could be used to evaluate software reliability.

SECTION 2. DETAILS OF INVESTIGATION

Software maturity and software reliability are not equivalent. Software reliability is an important characteristic of software maturity. Software maturity is "a measure of the software's evolution toward satisfying all documented user requirements." Reliability is one such requirement. Maintainability, integrity, and portability are examples of others. So software reliability is a part of software maturity. With this important distinction between software maturity and software reliability in mind, a set of software reliability models was identified for use in estimating software reliability for assessing software maturity. Software reliability measures are not the only measures of software maturity. Available reliability estimation methods from industry, academia, and government agencies were first identified through a survey that involved attendance at appropriate seminars, examination of the literature, and consultation with other organizations. After the survey, the available reliability estimation methods were then evaluated using several criteria. Of all the reliability estimation methods surveyed, only the available software reliability models were found to satisfy all of the criteria. They were found to eliminate the guesswork from system testing through the application of statistical analysis, the need for which is widely recognized in all fields of serious scientific endeavor.

2.1 SURVEY OF SOFTWARE RELIABILITY MODELS AND METHODOLOGIES.

A set of automated software reliability models was identified through a survey of existing software reliability estimation methods. The survey was conducted in the following ways: conferences on software reliability and testing were attended to see firsthand what software reliability estimation methods are currently available; government agencies and academia were consulted; and a search of the literature was performed.

2.1.1 Available Software Reliability Models.

A software reliability model is a mathematical expression that can be used to quantify/predict the failure behavior of software. Such models must consider pertinent factors that affect software reliability such as the following: fault introduction, fault removal, and software operational environment. A good software reliability has the following major characteristics: it gives good predictions of future failure behavior; it computes useful quantities; it is simple to interpret and understand; it is widely acceptable; and, it is based on sound assumptions. The software reliability models that are readily available to the United States Army Electronic proving Ground (USAEPG) include the SPPA and the set of models contained within the SMERFS. SPPA was developed by USAEPG (reference 14). SMERFS was developed by the NSWC after their own extensive survey of software reliability estimation techniques (reference 7). SMERFS is a software package consisting of eight well-known models appearing in the literature.

2.1.1.1 Software Performance Parameter Assessment.

SPPA is a mathematical model which provides quantitative measures of software reliability. It was developed by USAEPG to meet the requirements of the old test and evaluation directive, DoDD 5000.3 (reference 18). That directive required a quantitative measure of software maturity for embedded computer software. This directive has been superseded by a new directive (reference 9) which still requires, through DoDD 5000.3-M-1 (reference 10), such quantitative measures.

2.1.1.1.1 SPPA Model Description.

The SPPA is a mathematical model of the fault appearance and removal process. It is an example of a population model which consists of a Markov or birth/death process. A population model possesses states and transitions between states. For a given state, a population consists of a number of individuals. A transition from one state to another corresponds to either an increase or decrease in population. For this application, a state corresponds to the number of active or discovered software faults and a transition corresponds to either the removal (death) of a software fault or the evocation (birth) of a software fault. For a detailed discussion of the mathematics involved in this model, see Volume II of the SPPA Methodology Investigation Final Report (reference 14).

2.1.1.1.2 SPPA Model Inputs.

The inputs to the SPPA model consist of the fault mean function and the repair mean function. Each of these functions have parameters which must be estimated before the functions themselves can be applied. The original evaluation of SPPA indicated that this is a problem area (reference 19). This evaluation recommended that this initial parameter estimation should be automated. Initial parameter estimation currently has to be done by hand calculation. These calculations, being tedious and time consuming, impose an unrealistic data requirement.

The fault mean function models the software fault counting process. This process is assumed to be Poisson in nature. The actual fault mean function can be described as the product of a Weibull distribution and a constant. The function is estimated from histogram data using techniques from numerical analysis. The specific techniques include maximization of a Poisson likelihood function and the method of constr ined generalized least squares.

The repair mean function models the software repair counting process. The parametric form of this function is also that of a Weibull distribution. Like the fault mean function, this function is estimated using the same techniques of numerical analysis.

One additional input includes the number of debuggers. The number of debuggers responsible for generation of program repairs or patches is assumed to be a known quantity. The original evaluation indicated the number of debuggers is an unrealistic if not impossible data requirement because the model does not allow for the fluctuation of this number over the course of testing.

2.1.1.1.3 SPPA Model Outputs.

The SPPA model outputs the following reliability estimates: mean time between events, mean time to next fault, number of faults remaining, length of successful mission, time to last fault, extinctions of the active fault population, mean active fault count, time to next extinction, time to last repair, and probability of zero faults.

2.1.1.2 SMERFS Interactive Software Package of Models.

SMERFS is an interactive software package which performs a software reliability analysis using any of eight well-known models appearing in the literature (reference 6). These models include: Littlewood and Verrall's Bayesian Reliability Growth Model, Moranda's Geometric Model, John Musa's Execution Time Model, an adaptation of Amrit Goel's NHPP Model, the Generalized Poisson Model, Amrit Goel's NHPP Model, Brooks and Motley's Discrete Software Reliability Model for a Software System, and Norman Schneidewind's Model. A discussion of each model's assumptions, inputs, and outputs is presented in Appendix D of this report. For an in-depth discussion, see the SMERFS User's Guide (reference 16).

2.1.1.2.1 Motivation for SMERFS.

There are several motivations for using SMERFS. One is that it eliminates the guesswork from the task of software reliability estimation by applying statistical analysis. SMERFS incorporates models which apply the approach that has received the greatest emphasis in the literature. That approach focuses upon modeling either the times between when errors are detected (measured in CPU or wall clock time) or the number of errors detected during each testing period.

SMERFS allows for a variety of modeling approaches such as exponential, Poisson, binomial, and Bayesian models which have demonstrated adaptability to handle a range of data sets. By making available a collection of models, the software analyst can run all the models and select the one which best fits the data set of the analyst.

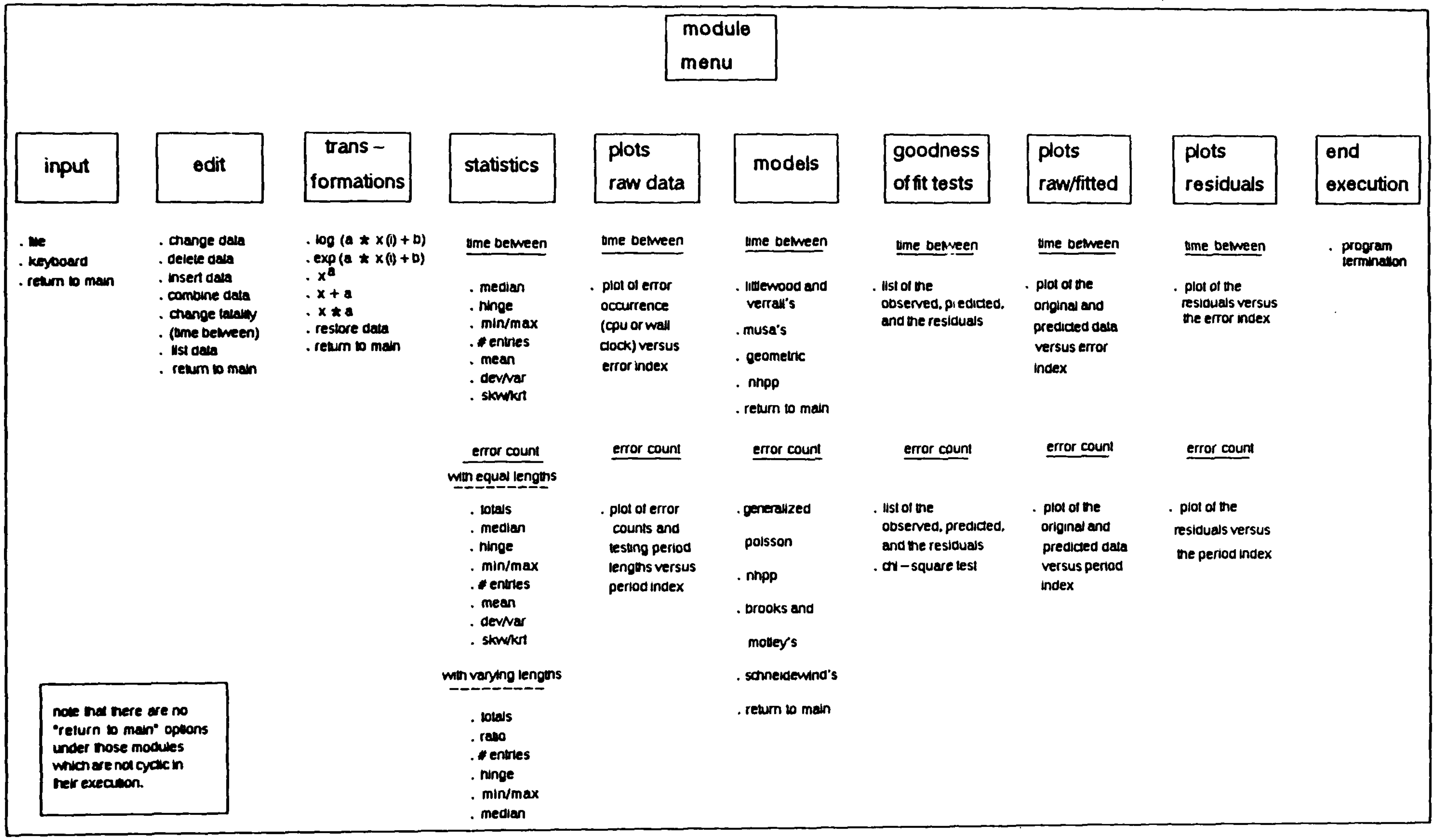
SMERFS provides a complete reliability analysis which is fully automated. It performs the difficult process of estimating the parameters of the distributions upon which the models are based. The parameter estimation process requires highly sophisticated numerical techniques which would be prohibitively time consuming, tedious, and hence, error prone if done by hand. The reliability analysis, furthermore, includes required statistical analyses which would be formidable to do by hand. These statistical analyses allow one to check the reasonableness of the model assumptions.

2.1.1.2.2 Features of SMERFS.

SMERFS has five desirable characteristics which can be summarized as follows: it is maintainable; it performs a complete reliability analysis; it is interactive; it detects user errors; and it is portable.

2.1.1.2.3 SMERFS Program Structure.

The basic SMERFS program structure is exhibited in Figure 2.1-1. SMERFS is a menu driven system which provides the user with various menus, and it prompts the user for inputs during execution (reference 6). The user inputs free-format responses via a terminal keyboard. The SMERFS User's Guide (reference 16) provides a detailed discussion of the prompts and menus.



Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

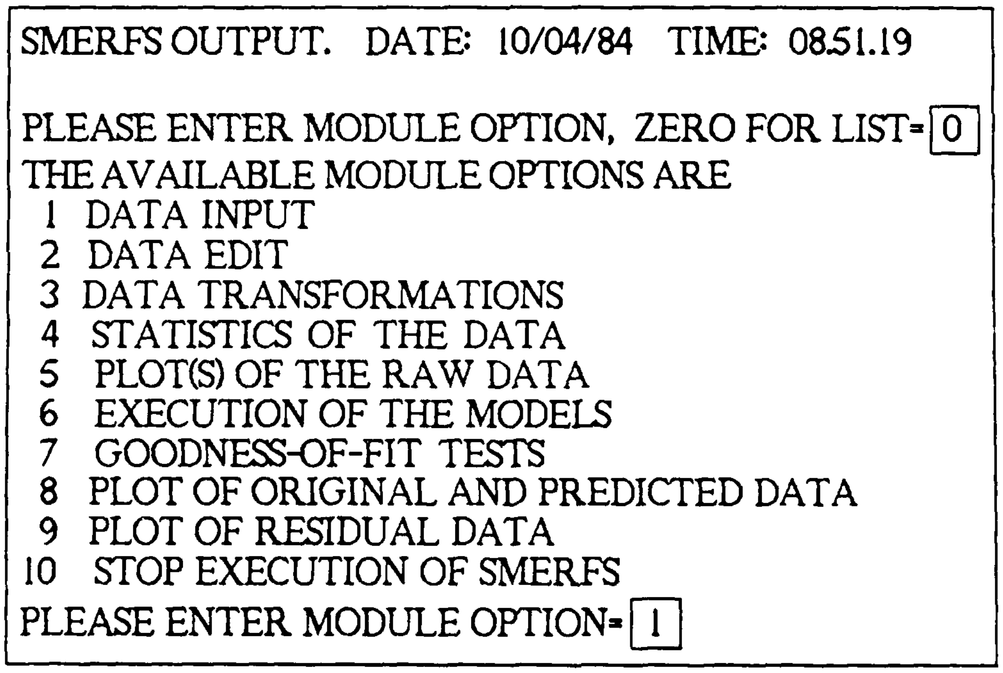
Figure 2.1 – 1. SMERFS Program Structure

2.1.1.2.4 Sample SMERFS Reliability Analysis.

An illustration of the use of the SMERFS program in performing a reliability analysis for a set of data is now provided. The original sample analysis, documented elsewhere (reference 6), was accomplished executing an earlier version of SMERFS. The latest version of SMERFS, which is Version III, differs very little from previous versions.

Like the sample analysis previously documented (reference 6), this example will not illustrate all of the SMERFS options such as data transformations and model fitting. For a more detailed description of all of the SMERFS options, consult either Appendix D of this document or see the SMERFS User's Guide (reference 16). Since the model chosen for this example is Goel's NHPP Model, none of the SMERFS features as applied to time between error detections are shown since Goel's NHPP Model does not use error count data.

During the execution of SMERFS, the main menu shown in Figure 2.1-2 appears on the terminal screen. The various module options are listed in the order in which one would want to perform an analysis.



```
SMERFS OUTPUT.   DATE: 10/04/84   TIME: 08.51.19

PLEASE ENTER MODULE OPTION, ZERO FOR LIST= [0]
THE AVAILABLE MODULE OPTIONS ARE
  1  DATA INPUT
  2  DATA EDIT
  3  DATA TRANSFORMATIONS
  4  STATISTICS OF THE DATA
  5  PLOT(S) OF THE RAW DATA
  6  EXECUTION OF THE MODELS
  7  GOODNESS-OF-FIT TESTS
  8  PLOT OF ORIGINAL AND PREDICTED DATA
  9  PLOT OF RESIDUAL DATA
 10  STOP EXECUTION OF SMERFS
PLEASE ENTER MODULE OPTION= [1]
```

NOTE: Blocked entries represent user input.

Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-2. SMERFS Main Menu

The data input option would be chosen first. This option allows the user to input actual data from the field to fit to a reliability model. Data input is exhibited in Figure 2.1-3. The available input options are file input or keyboard input. These options can be displayed through a menu. Once the input option is specified, data is entered through either a preexisting file if the file input option is selected or a terminal keyboard if the keyboard option is selected. In this example, the keyboard option is selected. Under this option, the program then prompts for the type of data to be entered. The SMERFS model data requirements can be seen in Table 1.5-III. Since Goel's NHPP Model uses error counts, the interval counts and lengths option is chosen in our example. As Figure 2.1-3 shows, once all of the data has been input, SMERFS prompts the user to return to the main menu to pick the next module option.

If a data entry error occurs, data can be edited at this point by selecting the data edit option. Data can also be transformed by selecting the data transformation option. SMERFS provides numerous transformations with which to do this. The options for editing and transforming data can be seen in Figure 2.1-1.

The user can select the option called "Statistics of the Data" to obtain summary statistics on the data that was entered (Figure 2.1-4).

These statistics include the median error count and the number of errors found up to this point. These statistics also include the following measures for the data: the mean, standard deviation, variance, and the coefficients of skewness and kurtosis.

In this continuing example, the fifth module option is now chosen to obtain plots of the raw data (Figure 2.1-5). This results in two plots of data. One plot shows raw counts of errors per testing period versus the number of the testing period. The other plot charts testing period length versus testing period number.

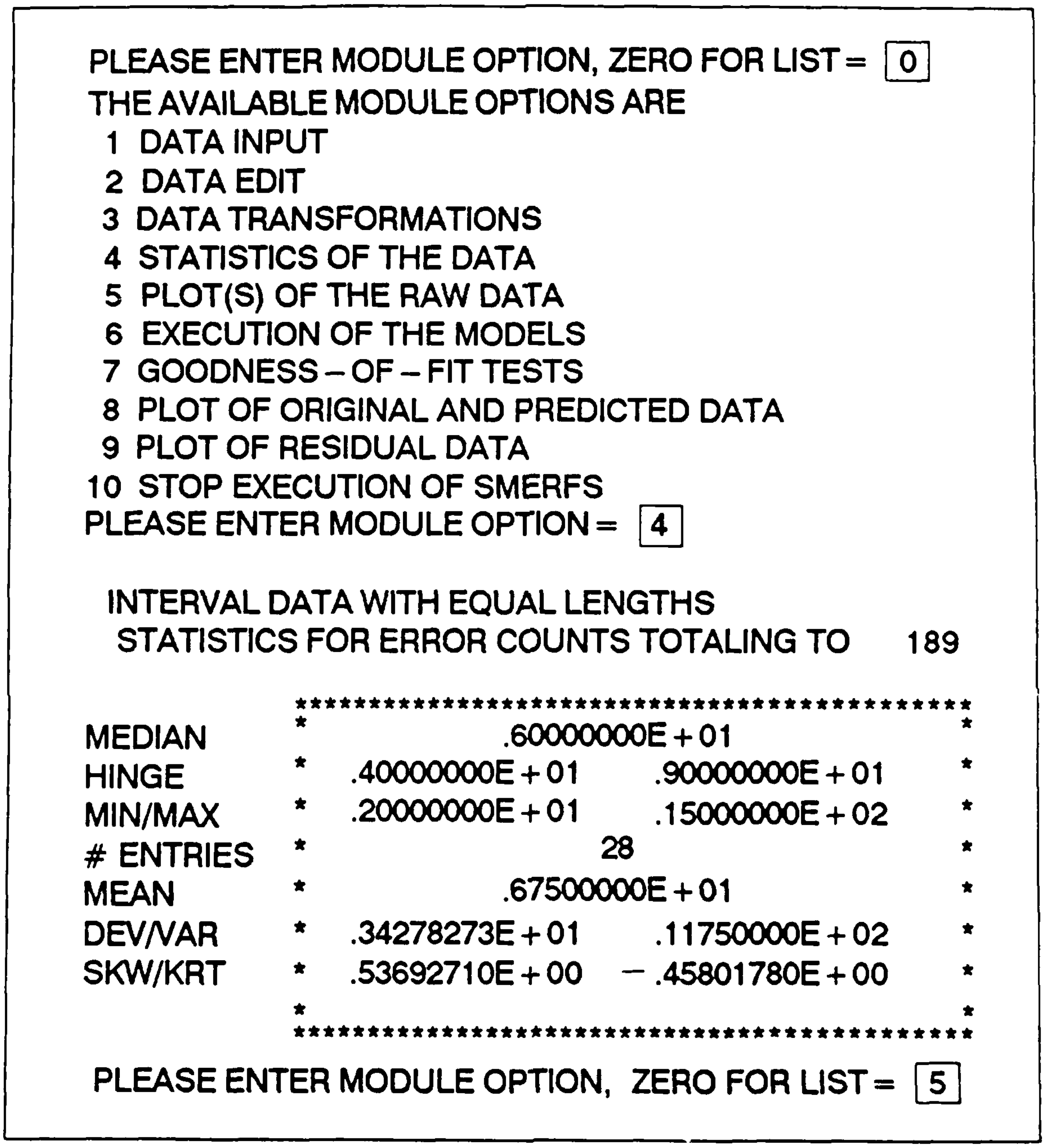
The "Execution Of The Models" option is chosen next to select the model devised to fit to the data. Figure 2.1-6 illustrates the menus and prompts at this step. For this example, Goel's NHPP model was selected. A list of the model assumptions and data requirements can be provided to the user at this point to enable him or her to decide on the model's applicability. If the user decides to continue with this candidate model, prompts request inputs needed to determine the parameters of the model. The inputs at this point consist of initial estimates of the number of iterations to be used by the model estimation procedures and initial guesses at model parameters. The number of iterations to be used in numerical procedures implemented by the model must be chosen so these procedures will converge to a solution. If successful convergence occurs, reliability estimates and their corresponding precision are output; otherwise, the user will have to try a larger number of iterations or a more appropriate initial guess for model parameters. Figure 2.1-7 exemplifies these model estimation procedures for our continuing example.

```
PLEASE ENTER INPUT OPTION,  ZERO FOR LIST =   0
THE AVAILABLE INPUT OPTIONS ARE
  1  FILE INPUT
  2  KEYBOARD INPUT
  3  RETURN TO THE MAIN PROGRAM
PLEASE ENTER INPUT OPTION =   2
PLEASE ENTER KEYBOARD OPTION,  ZERO FOR LIST =   0
THE AVAILABLE KEYBOARD INPUT OPTIONS ARE
  1  WALL CLOCK TIME - BETWEEN - ERROR  (WC TBE)
  2  CENTRAL PROCESSING UNITS (CPU)  TBE
  3  WC TBE AND CPU TBE
  4  INTERVAL COUNTS AND LENGTHS
  5  RETURN TO THE INPUT ROUTINE
PLEASE ENTER KEYBOARD INPUT OPTION =   4
  A RESPONSE OF NEGATIVE VALUES FOR THE PROMPT
  "PLEASE ENTER ERROR COUNT AND TEST LENGTH ="
  WILL STOP PROCESSING

  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    9    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =   15    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    9    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =   13    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    9    1

                        ●
                        ●
                        ●

  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    3    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    3    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    3    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =    5    1
  PLEASE ENTER ERROR COUNT AND TEST LENGTH =   -1   -1

PLEASE ENTER INPUT OPTION,  ZERO FOR LENGTH =   3
```

NOTE: Blocked entries represent user input.

Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-3. SMERFS Data Input

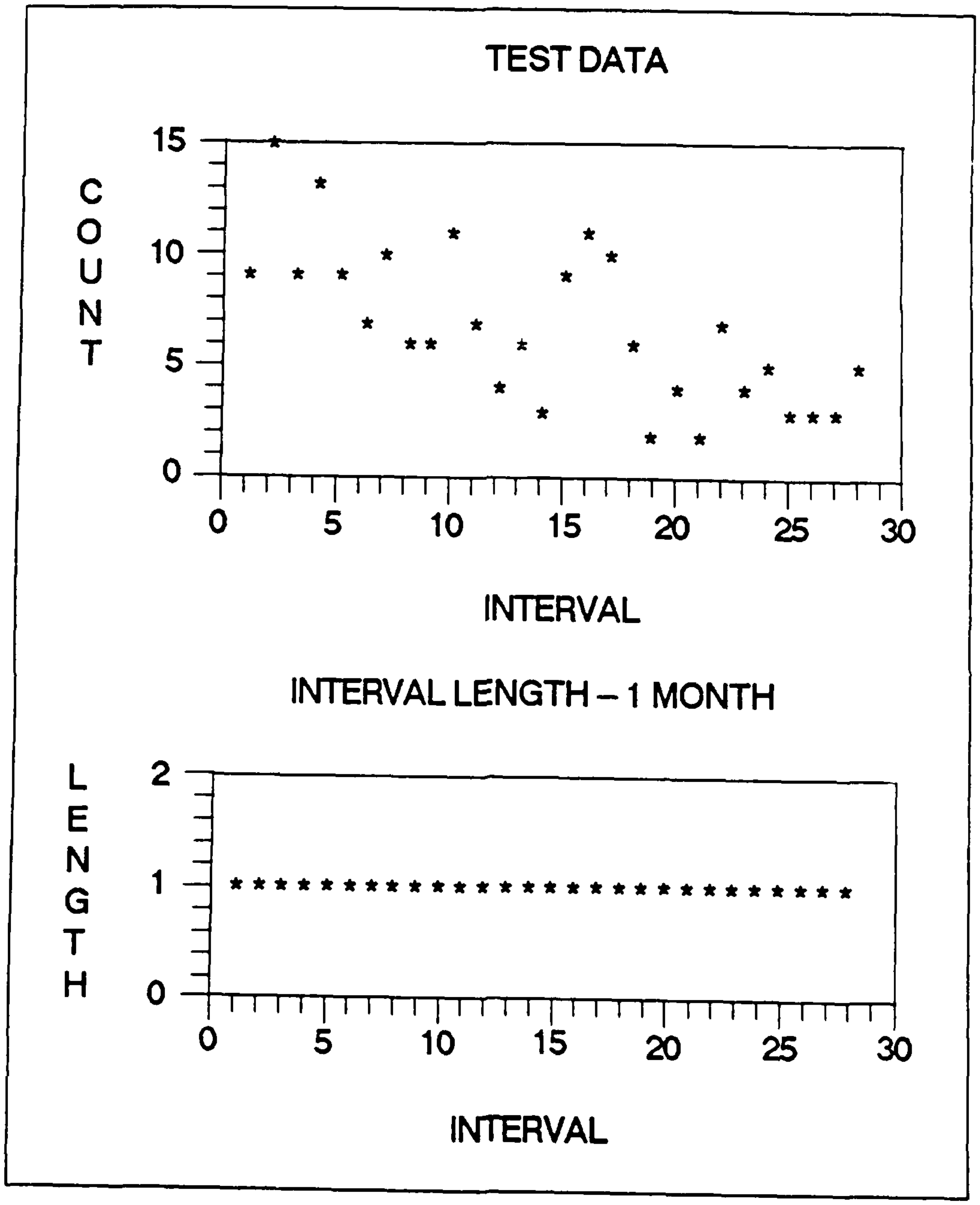


```
PLEASE ENTER MODULE OPTION, ZERO FOR LIST = 0
THE AVAILABLE MODULE OPTIONS ARE
  1  DATA INPUT
  2  DATA EDIT
  3  DATA TRANSFORMATIONS
  4  STATISTICS OF THE DATA
  5  PLOT(S) OF THE RAW DATA
  6  EXECUTION OF THE MODELS
  7  GOODNESS-OF-FIT TESTS
  8  PLOT OF ORIGINAL AND PREDICTED DATA
  9  PLOT OF RESIDUAL DATA
 10  STOP EXECUTION OF SMERFS
PLEASE ENTER MODULE OPTION = 4

  INTERVAL DATA WITH EQUAL LENGTHS
   STATISTICS FOR ERROR COUNTS TOTALING TO      189

            **********************************************
MEDIAN      *              .60000000E+01                 *
HINGE       *   .40000000E+01        .90000000E+01       *
MIN/MAX     *   .20000000E+01        .15000000E+02       *
# ENTRIES   *                28                          *
MEAN        *              .67500000E+01                 *
DEV/VAR     *   .34278273E+01        .11750000E+02       *
SKW/KRT     *   .53692710E+00       -.45801780E+00       *
            *                                            *
            **********************************************

 PLEASE ENTER MODULE OPTION,  ZERO FOR LIST = 5
```

NOTE: Blocked entries represent user input.

Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-4. SMERFS Summary Statistics



Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

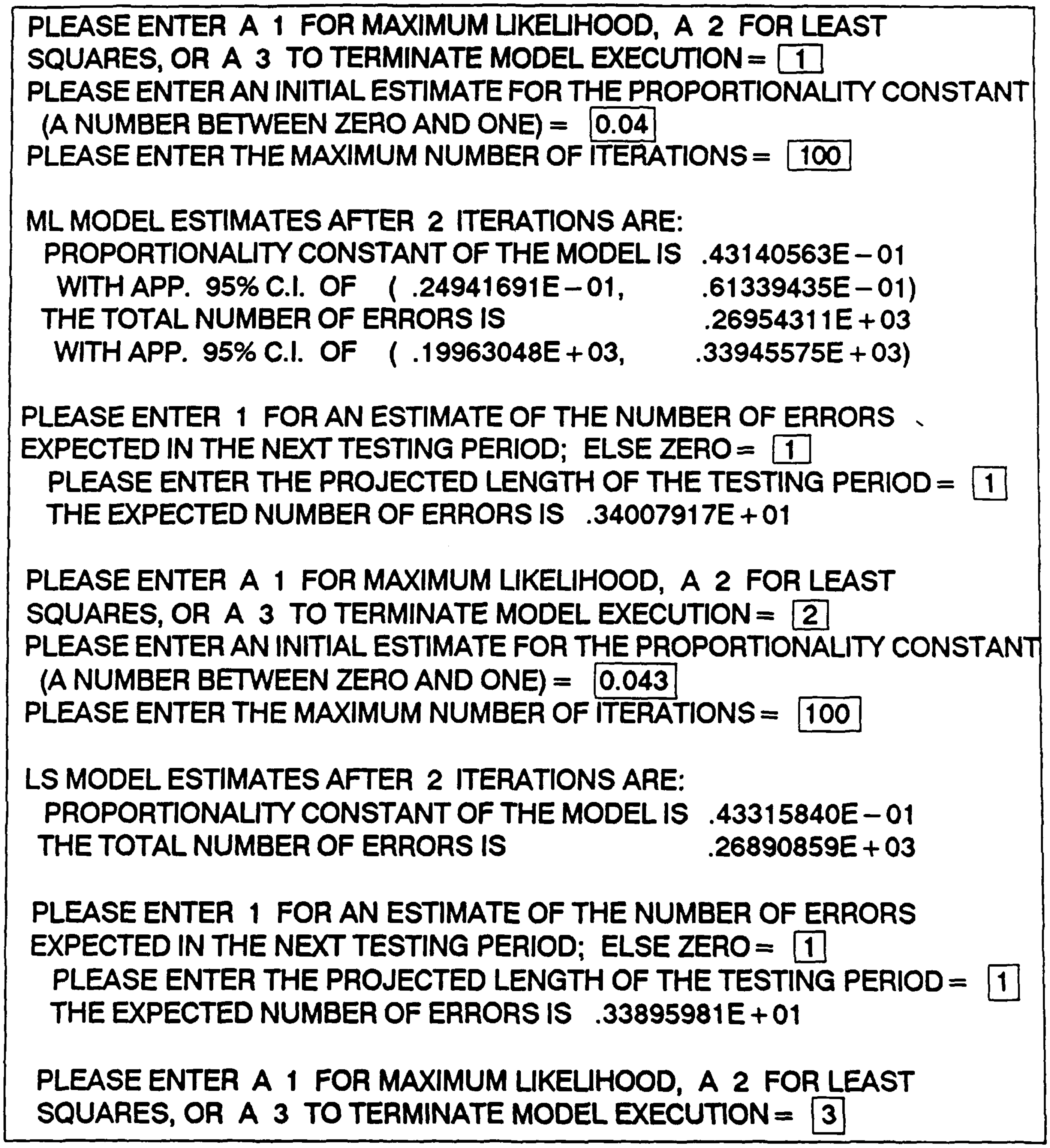
Figure 2.1-5. SMERFS Plots of the Raw Data

```
PLEASE ENTER MODULE OPTION,  ZERO FOR LIST = [0]
THE AVAILABLE MODULE OPTIONS ARE
  1  DATA INPUT
  2  DATA EDIT
  3  DATA TRANSFORMATIONS
  4  STATISTICS OF THE DATA
  5  PLOT(S) OF THE RAW DATA
  6  EXECUTION OF THE MODELS
  7  GOODNESS-OF-FIT TESTS
  8  PLOT OF ORIGINAL AND PREDICTED DATA
  9  PLOT OF RESIDUAL DATA
 10  STOP EXECUTION OF SMERFS
PLEASE ENTER MODULE OPTION = [6]

PLEASE ENTER COUNT MODEL OPTION,  ZERO FOR LIST = [0]
THE AVAILABLE ERROR COUNT MODELS ARE
  1  GENERALIZED POISSON MODEL
  2  NON-HOMOGENEOUS POISSON MODEL
  3  BROOKS AND MOTLEY'S MODEL
  4  SCHNEIDEWIND'S MODEL
  5  RETURN TO THE MAIN PROGRAM
PLEASE ENTER MODEL OPTION = [2]
```

NOTE: Blocked entries represent user input.

Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-6. SMERFS Execution of the Models



```
PLEASE ENTER A 1 FOR MAXIMUM LIKELIHOOD, A 2 FOR LEAST
SQUARES, OR A 3 TO TERMINATE MODEL EXECUTION = [1]
PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY CONSTANT
 (A NUMBER BETWEEN ZERO AND ONE) = [0.04]
PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS = [100]

ML MODEL ESTIMATES AFTER 2 ITERATIONS ARE:
  PROPORTIONALITY CONSTANT OF THE MODEL IS  .43140563E-01
   WITH APP. 95% C.I. OF   ( .24941691E-01,  .61339435E-01)
  THE TOTAL NUMBER OF ERRORS IS             .26954311E+03
   WITH APP. 95% C.I. OF   ( .19963048E+03,  .33945575E+03)

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS
EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO = [1]
  PLEASE ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD = [1]
  THE EXPECTED NUMBER OF ERRORS IS  .34007917E+01

PLEASE ENTER A 1 FOR MAXIMUM LIKELIHOOD, A 2 FOR LEAST
SQUARES, OR A 3 TO TERMINATE MODEL EXECUTION = [2]
PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY CONSTANT
 (A NUMBER BETWEEN ZERO AND ONE) = [0.043]
PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS = [100]

LS MODEL ESTIMATES AFTER 2 ITERATIONS ARE:
  PROPORTIONALITY CONSTANT OF THE MODEL IS  .43315840E-01
  THE TOTAL NUMBER OF ERRORS IS             .26890859E+03

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS
EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO = [1]
  PLEASE ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD = [1]
  THE EXPECTED NUMBER OF ERRORS IS  .33895981E+01

PLEASE ENTER A 1 FOR MAXIMUM LIKELIHOOD, A 2 FOR LEAST
SQUARES, OR A 3 TO TERMINATE MODEL EXECUTION = [3]
```

NOTE: Blocked entries represent user input.

Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-7. SMERFS Model Estimation Procedures

The user can determine the adequacy of the model by performing statistical analysis through SMERFS. One may perform a Chi-square goodness-of-fit test and see tables of the original, predicted, and residual data by choosing option 7. Figure 2.1-8 shows the Chi-square statistic and the tabulated data for this example. Figure 2.1-9 depicts the raw and fitted model together. Figure 2.1-10 shows a residual plot of the NHPP fit. The model fit of data and the residual plots are obtained through options 8 and 9, respectively. If the user determines that the model is inadequate based on these options, a different model will be fitted to the data. Alternately, the user could edit the data if it is found to be suspect and run the model again or try a different data transformation.

2.1.2 Other Approaches and Methodologies.

An attempt was made to identify approaches other than software reliability models which can estimate software reliability to assess software maturity. No such approach was found. Other methods which assess software maturity were identified, but none of them employ statistical analysis to compute reliability metrics such as mean time to failure (MTTF) and remaining number of software faults.

2.1.2.1 Alternative Approach.

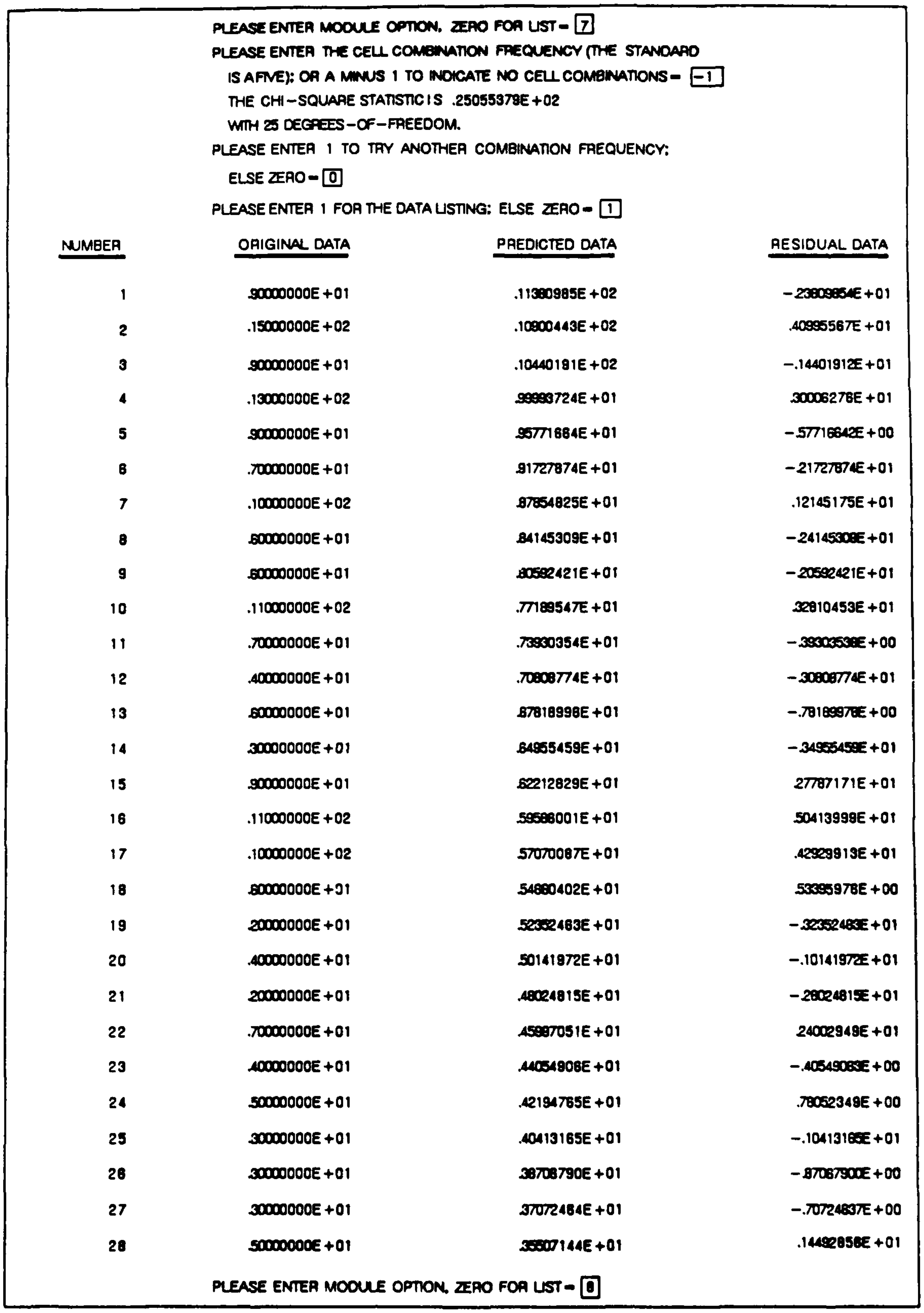
One approach developed by WSMR makes no attempt to measure software reliability (reference 11). Instead, it attempts to measure software maturity which is defined as "the software state of readiness to proceed to the next stage of its development." Many factors are considered in assessing software maturity, but not software reliability. The factors considered include results from various assessments such as extent of test, software change analysis, software performance assessment, and software test bed assessment. These assessments involve qualitative guidelines. The final determination of the maturity of software is done by making an engineering decision based upon the results of these assessments (reference 11). According to the creator of this approach, its methodology is not formally documented (reference 20).

2.1.2.1.1 Extent of Test Assessment.

The Extent of Test (EOT) assessment involves the determination of the extent to which each software requirement has been tested. Results are categorized in increasing order of desirability as either NOT TESTED, LIMITED, EXERCISED, or STRESSED. Alternatively, the ratio of the number of test conditions observed to the number of test conditions required can indicate the EOT.

2.1.2.1.2 Software Change Analysis.

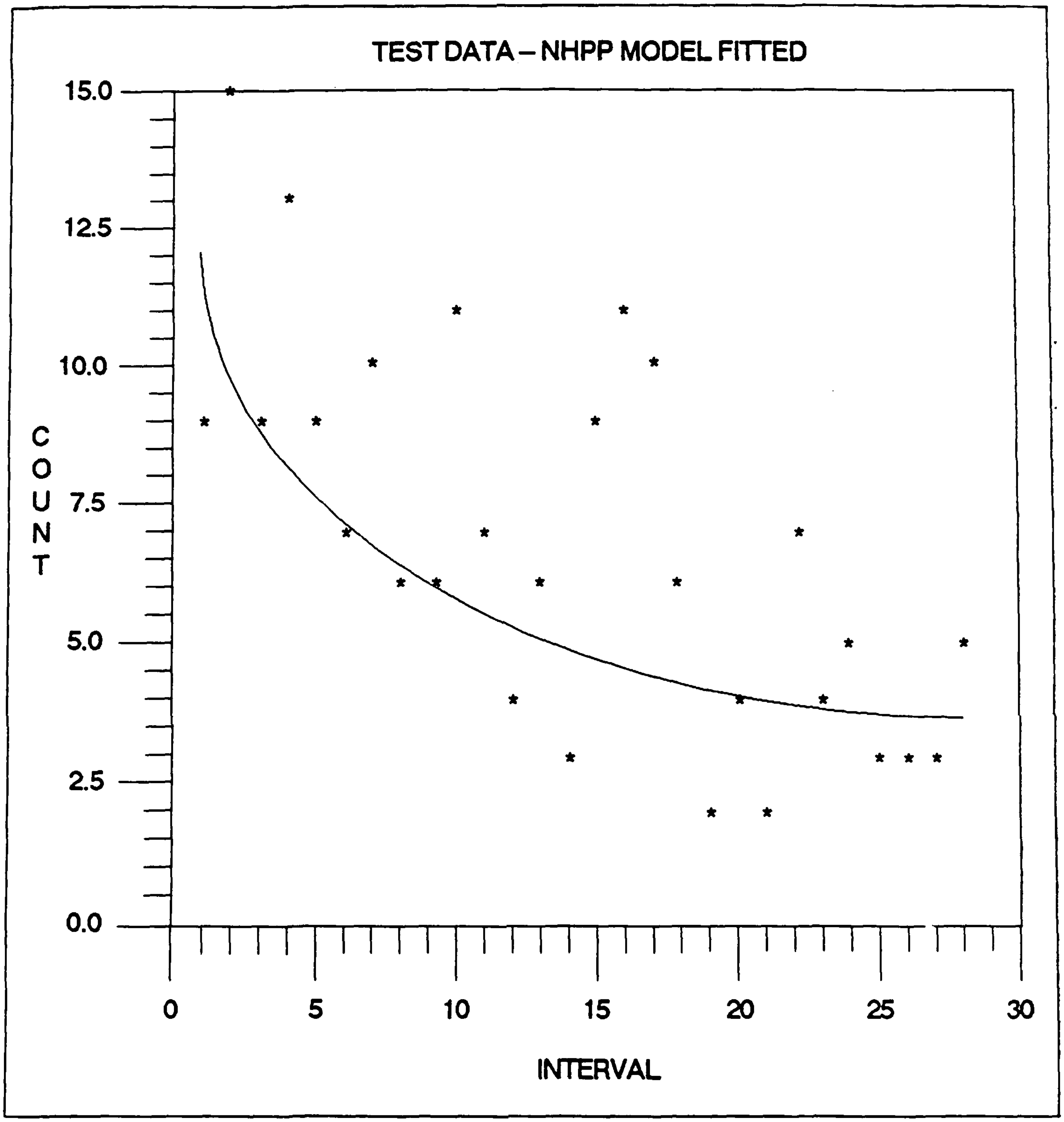
This involves a determination of the validity of software changes and the effect of software changes on the validity of previously obtained test results. A change profile which distinguishes changes due to requirements, design, and code is maintained. A trend analysis of this profile provides input to the assessment of software maturity.



```
PLEASE ENTER MODULE OPTION. ZERO FOR LIST= [7]
PLEASE ENTER THE CELL COMBINATION FREQUENCY (THE STANDARD
  IS A FIVE); OR A MINUS 1 TO INDICATE NO CELL COMBINATIONS= [-1]
  THE CHI-SQUARE STATISTIC IS .25055379E+02
  WITH 25 DEGREES-OF-FREEDOM.
PLEASE ENTER 1 TO TRY ANOTHER COMBINATION FREQUENCY;
  ELSE ZERO= [0]
PLEASE ENTER 1 FOR THE DATA LISTING; ELSE ZERO= [1]

NUMBER        ORIGINAL DATA         PREDICTED DATA          RESIDUAL DATA

   1         .90000000E+01         .11380985E+02         -.23809854E+01
   2         .15000000E+02         .10900443E+02          .40995567E+01
   3         .90000000E+01         .10440191E+02         -.14401912E+01
   4         .13000000E+02         .99993724E+01          .30006276E+01
   5         .90000000E+01         .95771664E+01         -.57716642E+00
   6         .70000000E+01         .91727874E+01         -.21727874E+01
   7         .10000000E+02         .87854825E+01          .12145175E+01
   8         .60000000E+01         .84145309E+01         -.24145308E+01
   9         .60000000E+01         .80592421E+01         -.20592421E+01
  10         .11000000E+02         .77189547E+01          .32810453E+01
  11         .70000000E+01         .73930354E+01         -.39303538E+00
  12         .40000000E+01         .70808774E+01         -.30808774E+01
  13         .60000000E+01         .67818998E+01         -.78189978E+00
  14         .30000000E+01         .64955459E+01         -.34955459E+01
  15         .90000000E+01         .62212829E+01          .27787171E+01
  16         .11000000E+02         .59586001E+01          .50413999E+01
  17         .10000000E+02         .57070087E+01          .42929913E+01
  18         .60000000E+01         .54660402E+01          .53395978E+00
  19         .20000000E+01         .52352483E+01         -.32352483E+01
  20         .40000000E+01         .50141972E+01         -.10141972E+01
  21         .20000000E+01         .48024815E+01         -.28024815E+01
  22         .70000000E+01         .45997051E+01          .24002949E+01
  23         .40000000E+01         .44054906E+01         -.40549063E+00
  24         .50000000E+01         .42194765E+01          .78052349E+00
  25         .30000000E+01         .40413165E+01         -.10413165E+01
  26         .30000000E+01         .38708790E+01         -.87087900E+00
  27         .30000000E+01         .37072484E+01         -.70724837E+00
  28         .50000000E+01         .35507144E+01          .14492856E+01

PLEASE ENTER MODULE OPTION. ZERO FOR LIST= [8]
```

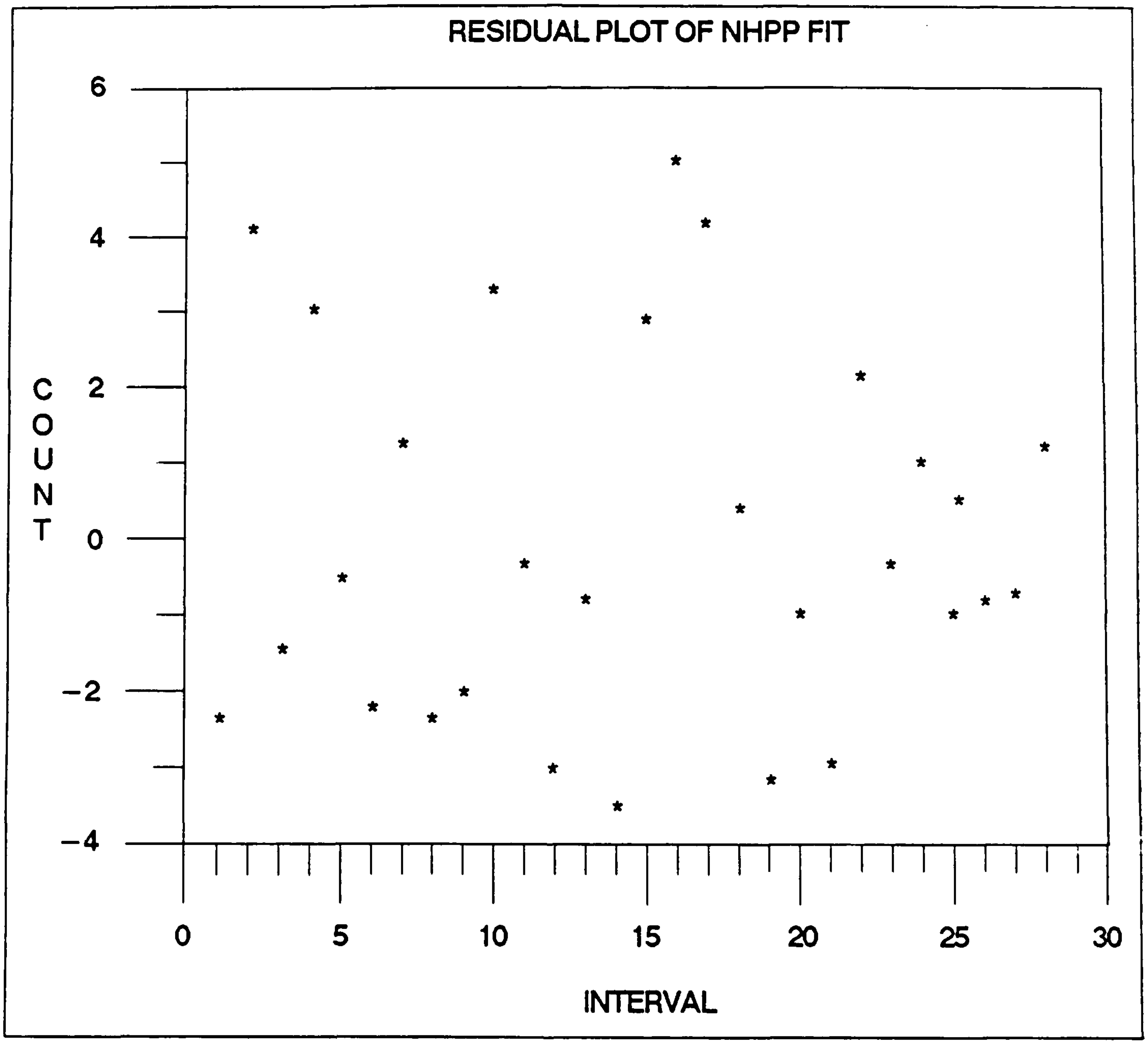
Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-8. SMERFS Chi-Square Statistic and Tabulated Data



Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-9. SMERFS Model Fit of Data



Source: An Interactive Program for Software Reliability Modeling, Farr, W.H., Smith, O.D., Naval Surface Warfare Center

Figure 2.1-10. SMERFS Plot of Residuals

2.1.2.1.3 Software Performance Assessment.

The software performance assessment determines what requirements have been met. In addition, it ascertains the boundaries and limitations of the capabilities of the software. These determinations provide a confidence indicator of the maturity of the software.

2.1.2.1.4 Other Factors Affecting Software Maturity.

Other factors include adequacy of test beds, adequacy of data collection, and quality of documentation. The WSMR approach addresses only adequacy of test beds while recognizing that other factors exist.

Test bed assessment determines the effectiveness of various test beds. This provides insights into where test resources can be spent most effectively. Insights can be achieved by comparing the Completeness of Test (COT) from Test Planning to the EOT from Test Analysis.

COT can be assessed using a Test Coverage Matrix (TCM) which maps the total set of software requirements and associated test conditions to planned developer and Government tests. The test bed supporting a given phase of testing is effective if such a comparison is favorable.

2.1.2.2 Another Alternative Approach.

AFOTEC evaluates the software of a system with methodologies which support two test objectives (reference 12). One of these objectives is software maturity. AFOTECP 800-2 Volume 1 (reference 12) defines software maturity as "a measure of the software's evolution toward satisfying all documented user requirements." In this approach, the number and severity of changes required to meet documented user requirements is the primary indicator of mature software (reference 12). Software maturity assessment utilizes a software tracking methodology and a software fault analysis.

2.1.2.2.1 Software Fault Tracking Methodology.

The Deputy of Software Evaluation (DSE), software evaluators, and deputies for operations and logistics evaluation take part in system testing, review system test data, and look into system software related performance deficiencies. Software problems and enhancements are maintained in a watch list under documented procedures. A level of severity is assigned to watch list items under Operational Test and Evaluation (OT&E) guidelines provided in an OT&E data management plan. Test teams submit service reports on watch list items thought by the test team to warrant particular attention. Periodic meetings are conducted during testing to review and validate software problems. The severity of software problems is also reviewed and validated at such meetings. A data base of software problems is maintained using a small computer system.

2.1.2.2.2 Software Fault Analysis.

This analysis uses two indicators. The primary indicator is the slope of the graph of new software faults being discovered during test. Software problems are tracked by a severity point system. Severity points are accumulated and plotted against elapsed time as testing progresses and new fault data is collected. A decrease in the slope of this graph or curve over time is the primary indicator of software maturity.

2.1.2.3 AMC-P 70-14 Approach.

AMC-P 70-14 describes software quality indicators designed to provide program managers with an "early warning mechanism for detecting software quality problems before they reach the field" (reference 13). Three of these indicators can provide insight into the reliability and maturity of software.

2.1.2.3.1 Fault Density.

The fault density indicator for a Computer Software Configuration Item (CSCI) uses two metrics. One of these is the cumulative faults divided by the total number of Computer Software Units (CSUs) in the CSCI. The other is the cumulative faults corrected divided by the total number of CSUs in the CSCI.

AMC-P 70-14 gives a possible rule of thumb for using this indicator to assess software maturity. According to AMC-P 70-14, "Fault density should begin to level off during the midpoint of test and flatten as testing nears completion." The pamphlet points out that failure of the fault density curve to exhibit such characteristics "may be indicative of immature software."

2.1.2.3.2 Test Coverage.

The test coverage indicator is a measure of the completeness of testing progress from a developer and user perspective. The indicator is the product of the percentage of requirements implemented and the percentage of software structure tested. The percentage of requirements implemented is the ratio of the number of tested implemented capabilities to the total required capabilities. The percentage of software structure tested is the ratio of software structure tested to the total software structure. Software structure is a function of the level and depth of testing. Its inputs may be units, segments, statements, branches, or path test results.

2.1.2.3.3 Test Sufficiency.

The test sufficiency indicator assesses sufficiency of software integration and system testing based upon a prediction of the remaining software faults. Indicator inputs include the following: total number of faults predicted in the software; number of faults detected before software integration testing; number of units integrated; total number of units in the CSCI; and, total number of faults detected to date during test. The total faults predicted has to be estimated. AMC-P 70-14 indicates ways of doing this.

2.2 EVALUATION OF THE MODELS AND OTHER METHODS.

Upon completion of the survey, the available reliability estimation models and other methods were evaluated according to the following criteria:

a. Does the model or other method provide probabilistic and quantitative estimates of program reliability?

b. Is the model or other method sufficiently documented?

c. Does the model or other method have realistic data requirements?

d. Is the model or other method readily available?

e. Is the model or other method applicable to DT?

Only the software reliability models within the SMERFS were found to satisfy all of the criteria. Only these models perform statistical analysis to eliminate the guesswork from software maturity estimation.

2.2.1 Evaluation of SPPA.

SPPA has been previously evaluated (reference 19). That evaluation concluded that the data requirements of SPPA are unrealistic and that it is poorly documented. In that evaluation it was recommended that a different model be used to estimate software reliability.

In reevaluating the SPPA, the same conclusions were drawn. Although SPPA satisfies three of the necessary criteria, it fails to satisfy the criteria for realistic data requirements and sufficient documentation.

2.2.2 Evaluation of SMERFS.

SMERFS is a powerful reliability estimation tool. It has been applied with varying degrees of success by IBM on the Space Shuttle Program (reference 15), the United States Navy on the Trident Missile Program (reference 16), and Hughes Aircraft on a continental air defense system (reference 17). In all cases, SMERFS provided conservative estimates of software reliability.

SMERFS is well documented and is available at no cost for use on microprocessor systems. It performs a complete reliability analysis of software failure data. This reliability analysis involves statistical analysis which computes quantitative and probabilistic estimates of software reliability.

The TIR does not include sufficient information to run the SMERFS models. For example, TIRs do not provide CPU time expended between software failures; nor do TIRs provide the starting and ending times of testing. These data are needed to run the CPU time between failure models and the software failure count models, respectively. Until TIRs are upgraded to include information, this data will have to come from elsewhere. The data can be made available through test officer reports, for example.

2.2.3 Evaluation of Other Approaches.

None of the alternative approaches satisfied all of the criteria. None of these approaches, therefore, were selected as candidates for application to DT.

The one approach developed by WSMR is certainly applicable to DT; its data requirements, however, are not clear. This problem stems from the fact that the approach is not formally documented, which is a problem in itself.

Its greatest drawback, however, is that it completely ignores statistical analysis, which is used extensively in all fields of serious scientific endeavor. Statistical analysis takes the guesswork out of system testing. It enables one to provide probabilistic, quantitative estimates of software reliability. This is something which WSMR's system oriented approach does not do. For these reasons, this approach was eliminated as a way to estimate software reliability. This does not preclude its use for assessing software maturity. Software maturity encompasses software reliability. This approach just does not address the reliability factor of software maturity.

The one approach developed by AFOTEC satisfies all the requirements except one: it does not provide probabilistic and quantitative estimates of software reliability. Its software maturity indicator, given as the slope of a line, cannot tell us the expected time to the next software failure or the number of faults remaining in the software. Such metrics are necessary if we are really serious about wanting to know when to stop testing. The key words here are indicator and metric. Software metrics measure some property of software. Software indicators provide insight into software quality, but they do not really measure software quality. For this reason, this approach was eliminated as a way to estimate software reliability. This does not rule out its use in assessing software maturity. It is simply recommended that it not be used to estimate software reliability.

The AMC-P 70-14 approach was eliminated from candidacy for the same reason as the approach developed by AFOTEC. Like that approach, it can be used to help assess software maturity. This approach, however, does not perform reliability estimation. In fact, it requires input from reliability estimation.

This Page Intentionally Blank

APPENDIX A

METHODOLOGY INVESTIGATION PROPOSAL

A-1. Title. Software Maturity Model Validation.

A-2. Category. All Department of the Army (DA) mission areas for systems containing embedded computer resources (ECR) are supported.

A-3. INSTALLATION OR FIELD OPERATING ACTIVITY. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613-7110.

A-4. PRINCIPAL INVESTIGATOR. Mr. K. Van Karsen, Software and Interoperability Division, STEEP-ET-DS, AUTOVON 879-02090/2092.

A-5. STATEMENT OF THE PROBLEM. Essentially all systems being developed employ some use of computers and software. Unlike hardware, the metrics for "software Reliability, Availability and Maintainability (RAM)" are ill-defined. Hereafter, "maturity" will be used instead of RAM. DoD Directive 5000.3 requires a quantitative measure of software maturity; Developmental Test (DT) evaluators and testers are required to develop methods for determining software maturity. TECOM cannot quantitatively measure the maturity of the software embedded in computer driven systems.

A-6. BACKGROUND. A number of models have been developed to predict software maturity. However, none have been validated. Under TECOM project number 7-CO-RD9-EP1-004, USAEPG derived the Software Performance Parameter Assessment (SPPA), a mathematical model which provided estimates of software maturity. Unlike previous models, the SPPA took into consideration the repair process, wherein repairs need not be made directly after encountering a fault (bug). The SPPA model was used on data from Lipow and from the Position Location Reporting System (PLRS) project. A final report was submitted to Headquarters (HQ) TECOM and subsequently approved for distribution. Subsequent to the development of SPPA, new models have appeared, but have not been evaluated for applicability to the developmental testing environment. Also, some researchers have developed an integrated package of various models with the intent that one or more of the models would be appropriate for a given situation. USAEPG has acquired a government-owned package, courtesy of Naval Surface Warfare Center, containing eight different models. However, the suitability of these models with respect to the availability of required data has not been determined.

A-7. GOAL. To establish an accepted method for assessing the maturity of the software in ECR.

A-8. DESCRIPTION OF INVESTIGATION.

a. Summary. USAEPG will evaluate currently available software maturity models and propose the best for use in TECOM software testing.

b. Detailed Approach. The U.S. Army Electronic Proving Ground will:

(1) Phase I - First Year's Effort:

(a) Identify software maturity (reliability) models available from industry, academia, and government agencies.

(b) Examine the data requirements of the various models with respect to the data available during DT.

(2) Phase II - Second Year's Effort:

(a) Select a set of available models which meet the constraints imposed by data availability.

(b) Consult with cognizant individuals on the applicability of the candidate models to TECOM's test and evaluation mission, and select a final set of models for use during DT.

(c) Demonstrate the recommended methods by applying data from a selected tactical system, and evaluate the results.

c. Final Product(s).

(1) Phase I:

(a) A set of initial candidate software maturity models.

(b) Evaluation of the data requirements for application to DT.

(2) Phase II:

(a) Recommendations for a set of models to determine software maturity during DT.

d. Coordination. Coordination with TECOM activities will be accomplished through the TECOM Software Technical Committee (TSOTEC). Coordination with other organizations will be performed directly.

e. Environmental Impact Statement. Execution of this task will not have an adverse impact on the quality of the environment.

f. Health Hazard Statement. Execution of this task will not involve health hazards to personnel.

A-9. JUSTIFICATION.

a. Association with Mission. One of TECOM's missions is to perform developmental tests on ECR. The investigation is needed to advance the concept of software maturity. The Army Science Board report on testing of electronic systems, with emphasis on software intensive systems, advocates RAM (maturity) programs as essential.

b. Association with Methodology/Instrumentation Program. This project supports thrusts of the TECOM Methodology Program to improve the quality of testing as well as the test process. Instrumentation developed or acquired previously would be used to form the basis of instrumentation required by the methodology.

c. Present Capability, Limitations, Improvement and Impact on Testing if Not Approved.

(1) Present Capability. The current test capability provides information in the form of Test Incident Reports (TIRs), for assessing software maturity.

(2) Limitations. Appropriate maturity models have not been identified and validated for application to DT, even though some raw data (TIRs) are available for analysis. Most prior attempts to assess maturity have avoided the lack of a validated model by using rather crude methods. For example, maturity per DoD-STD-1679A is determined on the basis of the number and severity of unresolved software errors at the time of acceptance. The number of latent faults which may surface after deployment is not estimated.

(3) Improvement. USAEPG and other organizations have developed software maturity models which may be suitable for DT use. Identification and validation of a model which will work within the DT environment will greatly improve estimated maturity quality.

(4) Impact on Testing if Not Approved. The intent of DoDD 5000.3 is not met unless a quantitative means of evaluating maturity is provided. Reporting maturity as the amount of discovered faults, while ignoring latent faults, results in a distorted view of actual maturity, given the current test techniques.

d. Dollar Savings. No dollar savings can be assessed at this time. The potential of this project is that a quantitative measure of software maturity can be attained; provide insight to Program Manager's (PM's) and evaluator's as to the maturity of a given software system to prevent fielding of an immature system and the inherent high cost to fix once fielded.

e. Workload. Over the past 5 years, USAEPG has experienced 17 tests requiring an evaluation of software maturity.

Examples of items anticipated for testing include:

Test Item		Fiscal Year (FY) 88	89	90
MSE	(Mobile Subscriber Equipment)	X	X	X
JTIDS	(Joint Tactical Information Distribution System)	X		
MCS	(Maneuver Control System)	X	X	X
VISTA	(Very Intelligent Surveillance and Target Acquisition)	X	X	X
FADDC^2 I		X	X	X
JINTACCS	(Joint Interoperability of Tactical Command and Control Systems	X	X	X
EPLRS (formerly PJH)	(Enhanced Position Location Reporting System)	X	X	X
GPS	(Global Positioning System)	X		
ASAS	(All Source Analysis System)	X		
AFATDS	(Advanced Field Artillery Tactical Data System)	X	X	X

f. Association with Requirements Documents. DoD Directive 5000.3 requires a quantitative measure of software maturity for each development phase. To date, there are no accepted measuring schemes.

g. Others. N/A.

A-10. RESOURCES.

A. Financial.

Dollars (Thousands)

	FY88		FY89	
	In-House	Out-of-House	In-House	Out-of-House
Personnel Compensation	10.0		12.0	
Travel	2.0		3.0	
Contractual Support		52.0		45.0
Consultants & Other Svcs				
Materials & Supplies		1.0		5.0
Equipment				
General & Admin costs				
Subtotals	12.0	53.0	15.0	50.0
FY Totals	65.0		65.0	

b. Explanation of Cost Categories.

(1) Personnel Compensation. This cost represents compensation chargeable to the investigation for using technical or other civilian personnel assigned to the investigation.

(2) Travel. This represents cost incurred while visiting government and industry facilities.

(3) Contractual Support. Performance of the investigation will be accomplished with resources provided under an existing support contract.

(4) Consultants and Other Services. N/A.

(5) Material and Supplies. N/A.

(6) Equipment. N/A.

(7) General and Administrative Costs. N/A.

c. Obligation Plan.

	FY88				
Fiscal Quarter (FQ)	1	2	3	4	TOTAL
Obligation Rate (Thousands)	50.0	5.0	5.0	5.0	65.0

d. In-House Personnel.

(1) In-House Personnel Requirements by Speciality.

		Man-hours FY88 Only		
	Number	Required	Available	Total Required
Elect Engr, GS-0855	1	450	450	450

(2) Resolution of Non-Available Personnel. N/A

A-11. INVESTIGATION SCHEDULE.

	FY88												FY89											
	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S
In-House	-	.	-	.	-	.	-	.	-	.	-	I	-	.	-	.	-	.	-	.	-	.	-	R
Contracts Consultants	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Symbols:	----	Active investigation work (all categories)
	....	Contract monitoring (in-house only)
	I	Interim Report
	R	Final report due at HQ, TECOM

A-12. ASSOCIATION WITH TOP PROGRAM. TECOM Test Operations Procedure (TOP) 1-1-056, Software Testing, requires the assessment of software maturity. The results of this investigation may provide recommended changes to TOP 1-1-056 with regards to software maturity.

FOR THE COMMANDER:

ROBERT E. REINER
Chief, Modernization and
Advanced Concepts Division

This Page Intentionally Blank

APPENDIX B

REFERENCES

1. Software Reliability, Leone, A. M., Westinghouse Electric Corporation, Glen Burnie, Maryland, November 1988.

2. Software Reliability: Measurement, Prediction, Application, Musa, J. D., et al., McGraw-Hill, Inc., 1987.

3. RADC-TR-87-171, Methodology for Software Reliability Prediction, Volumes I-II, McCall, J., et al., November 1987.

4. RADC-TR-85-37, Volumes I-III, Specification of Software Quality Attributes, Bowen, T.P., et al., February 1985.

5. Methodology Investigation Final Report Specification Requirements for Software Evaluation, U. S. Army Electronic Proving Ground, Fort Huachuca, Arizona, July 1988.

6. An Interactive Program for Software Reliability Modeling, Farr, W. H., Smith, O. D., Naval Surface Warfare Center. Proceedings of the 9th Annual Software Engineering Workshop, NASA Goddard, SEL-84-0004, Maryland, 1984.

7. NSWC TR 82171, A Survey of Software Reliability Modeling and Estimation, Farr, W. H., September 1983.

8. NASA Contractor Report 4187, Quality Measures and Assurance for AI Software, Rushby, J., October 1988.

9. DoDD 5000.3, "Test and Evaluation," 1986.

10. DoD 5000.3-M-1, "Test and Evaluation Master Plan (TEMP) Guidelines," 1986.

11. A System-Oriented Methodology to Support Software Testability, Ellis, J. O. and Wygant, M. N., International Test and Evaluation Association (ITEA) Journal, Volume IX (1988), No. 2.

12. AFOTECP 800-2, Volume 1, Software Operational Test and Evaluation Guidelines, 1 August 1986.

13. AMC-P 70-14, Army Materiel Command Software Quality Indicators, 30 April 1987.

14. Methodology Investigation Final Report Software Performance Parameter Assessment Volumes I and II, U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona, 1981.

15. Onboard Primary Software Reliability Prediction (Space Shuttle Programs), Hamilton, D. O., Keller, T. W., IBM. Proceedings of the 11th Minnowbrook Workshop on Software Reliability, July 26 - 29 1988, Minnowbrook Conference Center, Blue Mountain Lake, New York.

16. NSWC TR 84-373, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) User's Guide, Farr, W. H. and Smith, O. D., April 1985.

17. Applicauion of a Multi-Model Approach to Estimating Residual Software Faults and Time Between Failures, Bowen, J. B., Quality and Reliability Engineering International, Volume 3, 41-51, 1987.

18. DoDD 5000.3, "Test and Evaluation," 1979.

19. SPPA Preliminary Review, Letter No. 83-171, Project No. 0310, 13 May 1983.

20. Telephone Conversation, Marthe Wygant, White Sands Missile Range, January 1989.

APPENDIX C

ACRONYMS AND ABBREVIATIONS

AFATDS.........	Advanced Field Artillery Tactical Data System
AFOTEC.........	Air Force Operational Test and Evaluation Center
AFOTECP........	Air Force Operational Test and Evaluation Center Pamphlet
AMC-P..........	Army Materiel Command - Pamphlet
APP............	Approximate
ASAS...........	All Source Analysis System
AUTOVON........	Automatic Voice Network
BAMMOD.........	Brooks and Motley Model
C^3I............	Command, Control, Communications, and Intelligence
COT............	Completeness of Test
CPU............	Central Processing Unit
CSCI...........	Computer Software Configuration Item
CSU............	Computer Software Unit
DA.............	Department of the Army
DARCOM.........	Department of the Army Readiness Command
DATINP.........	Data Input
DEV............	Standard Deviation
DoD............	Department of Defense
DoDD...........	Department of Defense Directive
DSE............	Deputy of Software Evaluation
DT.............	Developmental Testing
ECR............	Embedded Computer Resources
EOT............	Extent of Test
EXP............	Exponential
FQ.............	Fiscal Quarter
FY.............	Fiscal Year
GEOMOD.........	Geometric Model
GPOMOD.........	Generalized Poisson Model
GPS............	Global Positioning System
HQ.............	Headquarters
IBM............	International Business Machines Corporation
IOC............	Initial Operational Capability
ITEA...........	International Test and Evaluation Association
JINTACCS.......	Joint Interoperability of Tactical Command and Control Systems
JTIDS..........	Joint Tactical Information Distribution System
KRT............	Kurtosis
LAVMOD.........	Littlewood and Verrall Model
LOG............	Logarithm
LS.............	Least Squares
MAX............	Maximum
MCS............	Maneuver Control System
MIN............	Minimum
ML.............	Maximum Likelihood
MSE............	Mobile Subscriber Equipment
MTBF...........	Mean Time Between Failure
MTTF...........	Mean Time To Failure
MUSMOD.........	Musa Model

NHPP...........	Non-Homogeneous Poisson Process
NPIMOD.........	Non-Homogeneous Poisson Model
NPTMOD.........	Non-Homogeneous Poisson Execution Time Model
NSWC...........	Naval Surface Warfare Center
OCC............	Occurrence
OT&E...........	Operational Test and Evaluation
PJH............	PLRS Joint Hybrid
PLRS...........	Position Location Reporting System
PM.............	Program Manager
RADC...........	Rome Air Development Center
RAM............	Reliability, Availability, and Maintainability
SDWMOD.........	Schneidewind Model
SKW............	Skewness
SMERFS.........	Statistical Modeling and Estimation of Reliability Functions for Software
SPPA...........	Software Performance Parameter Assessment
TBE............	Time Between Error
TCM............	Test Coverage Matrix
T&E............	Test and Evaluation
TECOM..........	Test and Evaluation Command
TEMP...........	Test and Evaluation Master Plan
TFCS...........	Trident-I Fire Control System
TIR............	Test Incident Report
TOP............	Test Operations Procedure
TR.............	Technical Report
TSOTEC.........	TECOM Software Test and Evaluation Committee
USA............	United States Army
USAEPG.........	United States Army Electronic Electronic Proving Ground
VAR............	Variance
VISTA..........	Very Intelligent Surveillance and Target Acquisition
WC.............	Wall Clock
WSMR...........	White Sands Missile Range

APPENDIX D

SMERFS MODELS

D-1. GENERAL DISCUSSION.

A set of software reliability models for use in estimating software maturity is described below. The eight models are contained in the SMERFS interactive software reliability estimation package. Four of these models are time between failure models and four are error count models. The following information is provided for each model: model description, model assumptions, model inputs, model outputs. For more detailed information on the various prompts and options provided by these models, consult the SMERFS User's Guide and Farr's Survey of Software Reliability Modelling and Estimation. The time between and error count models are invoked by an execution time data model menu and an interval data model menu, respectively (Figures D-1.1 and D-1.2).

```
 PLEASE ENTER THE TIME MODEL OPTION, OR ZERO FOR A LIST.
 THE AVAILABLE WALL CLOCK OR CPU TIME MODELS ARE
  1 THE LITTLEWOOD AND VERRALL BAYESIAN MODEL
  2 THE MUSA EXECUTION TIME MODEL
  3 THE GEOMETRIC MODEL
  4 THE NHPP MODEL FOR TIME - BETWEEN - ERROR OCC.
  5 RETURN TO THE MAIN PROGRAM
 PLEASE ENTER THE MODEL OPTION.
IF WALL CLOCK AND CPU TBE DATA, THEN:
  PLEASE ENTER ONE FOR WC TBE OR TWO FOR CPU TBE.
    *** DATA TYPE ERROR; PLEASE TRY AGAIN (AFTER THE NEXT PROMPT).
END IF
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-1.1. Menu for Execution Time Models.

```
PLEASE ENTER THE COUNT MODEL OPTION,  OR ZERO FOR A LIST.
THE AVAILABLE ERROR COUNT MODELS ARE
  1  THE GENERALIZED POISSON MODEL
  2  THE NON – HOMOGENEOUS POISSON MODEL
  3  THE BROOKS AND MOTLEY MODEL
  4  THE SCHNEIDEWIND MODEL
  5  RETURN TO THE MAIN PROGRAM.
PLEASE ENTER THE MODEL OPTION.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-1.2. Menu for Interval Data Models.

D-2. THE LITTLEWOOD AND VERRALL BAYESIAN RELIABILITY GROWTH MODEL.

D-2.1. Model Description. Proposed by Littlewood and Verrall, this execution time data model tries to take into account the fact that the software correction process can introduce errors.

D-2.2. Model Assumptions. This model makes the following assumptions:

a. The software is operated in a manner similar to its expected operational usage.

b. Successive times between software failures are independent, exponentially distributed random variables x(i), i= 1,2,...,n with parameter μ(i).

c. The μ(i) are independent, Γ distributed variables with parameters α and π(i). α is a Γ function parameter. π(i) is a function which describes a programmer's quality and the programming task's difficulty. Littlewood and Verrall recommend a simple linear or quadratic function for the form of π. This recommendation is implemented in SMERFS.

D-2.3. Model Inputs. The model inputs include data entered via the SMERFS data input module, DATINP, and responses to prompts from the SMERFS LAVMOD module.

D-2.3.1. DATINP Inputs. The data input via the DATINP consists of the times between the error occurrences (i.e., the x(i)'s) measured in CPU or wall clock time. This is the raw data needed to run the model (i.e., the model data requirements).

D-2.3.2. LAVMOD Prompts. LAVMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-2.3.2.1. LAVMOD Description and List Prompts. SMERFS prompts the user through LAVMOD to see if the user wishes to see a list of the model's assumptions and data requirements. The model assumptions are those discussed above. The model data requirements are the inputs to DATINP.

D-2.3.2.2. LAVMOD Input and Prediction Vector Creation Prompts. The LAVMOD input prompts are exhibited in the menu in Figure D-2.1. The first prompt in that menu lets the user specify the desired method of estimating α, the Γ function parameter, and the linear or quadratic coefficients of the π function. The two methods of estimation allowed are maximum likelihood and least squares. The second prompt allows the user to specify whether he or she wants a linear or quadratic π function. The third prompt lets the user enter initial estimates for the linear or quadratic coefficients known as the β parameters. The final prompt lets the user enter the number of iterations to perform to obtain the maximum likelihood or least squares estimates of the α and β parameters.

D-2.4. Model Outputs. If successful convergence is achieved, LAVMOD outputs the expected mean time before the next error; otherwise, it lets the user try a larger number of iterations. In either case, estimates for the α and β parameters for the π function discussed above are output. The LAVMOD successful convergence output menu is seen in Figure D-2.2.

```
PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST
  SQUARES, OR 3 TO TERMINATE MODEL EXECUTION.

WHICH OF THE FOLLOWING FUNCTIONS DO YOU DESIRE
  TO USE AS THE PHI(I) IN THE GAMMA DISTRIBUTION?
  THE GAMMA IS USED AS THE PRIOR WITH PARAMETERS
  ALPHA AND PHI(I)

     1. PHI(I) = BETA(O) + BETA(1) * I (LINEAR)
  OR
     2. PHI(I) = BETA(O) + BETA(1) * I**2 (QUADRATIC).

PLEASE ENTER INITIAL ESTIMATES FOR BETA(O) AND BETA(1).

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-2.1. LAVMOD Input Prompts.

```
__MODEL ESTIMATES AFTER__ITERATIONS ARE:
     ALPHA    : ______________
     BETA(0)  : ______________
     BETA(1)  : ______________

THE FUNCTION EVALUATED AT THESE POINTS IS______________

PLEASE ENTER 1 FOR AN ESTIMATE OF THE MEAN TIME BEFORE
 THE NEXT ERROR; ELSE ZERO.

 THE EXPECTED TIME IS______________
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-2.2. LAVMOD Successful Convergence Output.

D-3. JOHN MUSA'S EXECUTION TIME MODEL.

D-3.1. Model Description. This execution time data model is based upon the amount of CPU time used in testing rather than upon the amount of wall clock or calendar time. In addition to modeling software reliability, this model can be used to model allocation of resources for testing segments and relate CPU time to wall clock time. The model is important for this reason.

D-3.2. Model Assumptions. The following assumptions are those needed only for reliability modeling. The assumptions for modeling resource allocation are documented in the SMERFS User's Guide.

a. The software is operated in a way similar to its expected operational usage.

b. The probability of detecting any given error is in no way affected by the occurrence of detecting another error (i.e., error detections are independent).

c. Every failure of software is observed.

d. The execution times between software failures are piecewise exponentially distributed. That is, the hazard rate function is a constant which changes whenever an error is corrected.

e. The ratio of the hazard rate to the number of errors remaining in the program is a constant.

f. The ratio of the rate of fault correction to the rate of failure occurrence is a constant.

D-3.3. Model Inputs The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS MUSMOD module.

D-3.3.1. DATINP Inputs. The data input via the DATINP module consists of the times between software failure occurrences measured in CPU time.

D-3.3.2. MUSMOD Prompts. MUSMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-3.3.2.1. MUSMOD Description and List Prompts. SMERFS prompts the user through MUSMOD to see if the user wishes to see a list of the model's assumptions and data requirements. The model assumptions are those discussed above. The model data requirements are the inputs to DATINP and the testing compression factor, C. This factor is the average ratio of the error detection rate during testing to that during operational use. This factor allows for changes in the operational environment.

D-3.3.2.2. MUSMOD Input and Prediction Vector Creation Prompts. The MUSMOD input prompts are exhibited in Figure D-3.1. The first prompt lets the user specify the testing compression factor. If there is no basis for estimation of this factor, a conservative approach would be to let C equal one. The second prompt allows the user to enter an initial estimate of the required number of software failures that must be experienced to uncover all software

```
PLEASE ENTER AN ESTIMATE FOR THE TESTING COMPRESSION
  FACTOR, C.
  IT IS THE AVERAGE RATE OF DETECTIONS OF ERRORS DURING
  THE TESTING PHASE TO THAT DURING USE. (A CONSERVATIVE
  VALUE IS 1.0).

PLEASE ENTER AN INITIAL ESTIMATE FOR THE TOTAL NUMBER
  OF ERRORS THAT MUST BE DETECTED IN ORDER TO UNCOVER
  ALL PROGRAM ERRORS.

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-3.1. MUSMOD Input Prompts.

faults within the program. The final prompt lets the user enter the maximum number of iterations to compute M, which is the required number of failures one needs to experience to uncover all faults within the program.

D-3.4. <u>Model Outputs</u>. If a solution is found before the maximum number of iterations is reached, successful convergence output occurs (Figure D-3.2). Otherwise, the user is allowed to repeat execution of the Musa model. After successful output, the user is prompted to see whether he or she wishes to run the Calendar Time Component. This component computes resource allocation for the testing segments. For further information on the description and outputs of the Musa Calendar Time Component, see the SMERFS User's Guide.

```
THE MAX. LIKELIHOOD ESTIMATES AFTER __ ITERATIONS ARE:

  1. THE TOTAL NUMBER OF ERRORS THAT MUST BE DETECTED BEFORE
     ALL ERRORS IN THE CODE ARE FOUND IS __________
     WITH APP. 95% C.I. OF (__________, __________)
  2. THE MAXIMUM LIKELIHOOD ESTIMATE OF THE INITIAL MEAN TIME
     BEFORE FAILURE (MTBF) FOR THE PROGRAM IS __________
     WITH APP. 95% C.I. OF (__________, __________)

THE ESTIMATE OF THE FAILURE MOMENT STATISTIC IS ______________
  WITH APP. 95% C.I. OF (______________, ______________)

THE ESTIMATE OF THE CURRENT MEAN TIME BEFORE THE NEXT
  SOFTWARE ERROR OCCURRENCE IS __________

AND THE ESTIMATE OF THE FUTURE RELIABILITY FOR THE SAME
  AMOUNT OF COMPLETED TESTING TIME IS ______________

PLEASE ENTER 1 TO ESTIMATE FUTURE RELIABILITY
  MEASURES AND TESTING TIME REQUIRED TO ACHIEVE
  SPECIFIED GOALS; ELSE ZERO.

  PLEASE ENTER THE DESIRED GOAL FOR MTBF.

  AN ADDITIONAL ______ ERRORS NEED TO BE DETECTED TO
  ACHIEVE THE DESIRED GOAL; AND THAT WILL CONSTITUTE AN
  ADDITIONAL ______ HOURS OF CPU TESTING TIME.

  PLEASE ENTER 1 TO TRY ANOTHER GOAL FOR MTBF;
  ELSE ZERO.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-3.2. MUSMOD Successful Convergence Output.

D-4. MORANDA'S GEOMETRIC MODEL.

D-4.1. Model Description. This execution time data model is a variation of the Jelinski-Moranda De-Eutrophication Model. The process of de-eutrophication presumes that the software hazard rate is reduced by the same amount at the time of each error detection. The software hazard rate is defined as the conditional probability that a software failure occurs in an interval of time given that the software has not failed up to the beginning of that time interval. The de-eutrophication process is geometric for this model because the hazard rate function decreases in a geometric progression as the detection of errors occurs.

D-4.2. Model Assumptions. The model presumes the following:

a. The software is operated in a way similar to its expected operational usage.

b. The program will never be error free.

c. The probability of detecting a given error may not equal the probability of detecting another given error.

d. The probability of detecting a given error is not affected by the probability of detecting another given error (i.e., the detection of errors is independent).

e. The rate at which errors are detected follows a geometric progression which is constant between error occurrences. This implies that errors become harder to detect as debugging progresses.

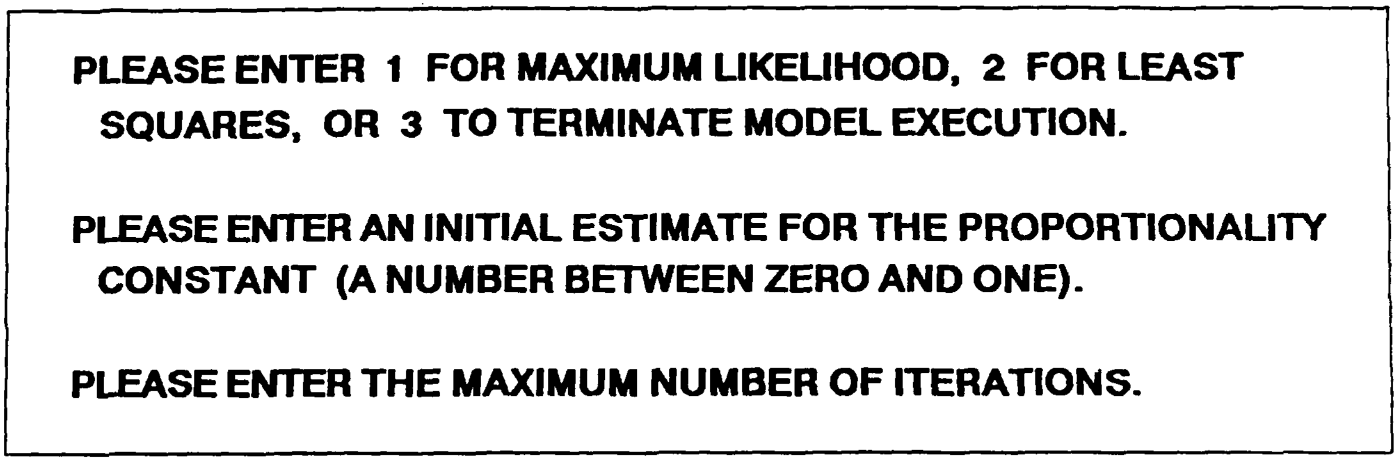
D-4.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS GEOMOD module.

D-4.3.1. DATINP Inputs. The data input via the DATINP module consists of the time between software failure occurrences measured in either CPU time or calendar (wall clock) time.

D-4.3.2. GEOMOD Prompts. GEOMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-4.3.2.1. GEOMOD Description and List Prompt. GEOMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-4.3.2.2. GEOMOD Input and Prediction Vector Creation Prompts. The GEOMOD input prompts are exhibited in Figure D-4.1. The first prompt lets the user terminate model execution or indicate the least squares or maximum likelihood method to estimate the proportionality constant for the software hazard function. The second prompt enables the user to enter an initial estimate for this constant. The user should choose a number between 0 and 1 to guarantee convergence of the solution. The final prompt allows the user to enter the maximum number of convergence iterations for the estimation technique chosen.



```
PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST
  SQUARES, OR 3 TO TERMINATE MODEL EXECUTION.

PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY
  CONSTANT (A NUMBER BETWEEN ZERO AND ONE).

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-4.1. GEOMOD Input Prompts.

D-4.4. Model Outputs. If a solution is found before the maximum number of iterations is reached, successful convergence output occurs (Figure D-4.2). Otherwise, the user is allowed to repeat execution of the model. As can be seen from Figure D-4.2, outputs include estimates for the proportionality constant, initial hazard rate, mean time before the next failure, and current purification level regardless of the estimation technique chosen (i.e., ML or LS). If maximum likelihood is chosen, it also provides 95 per cent confidence intervals for these estimates.

Since the model assumes infinite errors, it cannot compute the total number of errors in the program. Instead, it estimates the degree of "purification" for the program.

```
IF THE MAXIMUM LIKELIHOOD METHOD WAS SELECTED,  THEN:

   ML MODEL ESTIMATES AFTER ___ ITERATIONS ARE:
     PROPORTIONALITY CONSTANT OF THE MODEL IS  ____________
       WITH APP.  95%  C.I.  OF  (____________,  ______________)
     THE INITIAL HAZARD RATE IS                ______________
       WITH APP.  95%  C.I.  OF  (____________,  ______________)
     THE MEAN TIME BEFORE THE NEXT FAILURE IS  ______________
       WITH APP.  95%  C.I.  OF  (____________,  ______________)
     THE CURRENT  "PURIFICATION LEVEL"  IS     ______________
       WITH APP.  95%  C.I.  OF  (____________,  ______________)

ELSE,  IF THE LEAST SQUARES METHOD WAS SELECTED,  THEN:

   LS MODEL ESTIMATES AFTER ___ ITERATIONS ARE:
    PROPORTIONALITY CONSTANT OF THE MODEL IS  ____________
    THE INITIAL HAZARD RATE IS                ____________
    THE MEAN TIME BEFORE THE NEXT FAILURE IS  ____________
    THE CURRENT  "PURIFICATION LEVEL"  IS     ____________

END IF
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-4.2. GEOMOD Successful Convergence Output.

D-5. ADAPTATION OF GOEL'S NON-HOMOGENEOUS POISSON PROCESS MODEL.

D-5.1. Model Description. This execution time data model is an adaptation of Amrit Goel's NHPP interval count model.

D-5.2. Model Assumptions. This model's assumptions include the following:

a. The software is operated in a way similar its expected to operational usage.

b. The probability of detecting any given software error is the same as the probability of detecting any other given error.

c. The cumulative number of software errors detected up to a point in time are Poisson distributed. The expected number of software errors in any small interval of time $(t,t+\delta t)$ is proportional to the number of undetected software errors at time t.

d. The mean of the Poisson distribution, M(t), is a bounded non-decreasing function. As the length of testing tends to infinity, M(t) approaches the expected total number of eventually detected software errors.

D-5.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP module and response to prompts from the SMERFS NPIMOD module.

D-5.3.1. DATINP Inputs. The data input via the SMERFS DATINP consists of the time between software failure occurrences measured in either CPU time or calendar (wall clock) time.

D-5.3.2. NPIMOD Prompts. NPIMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-5.3.2.1. NPIMOD Description and List Prompts. NPIMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-5.3.2.2. NPIMOD Input and Prediction Vector Creation Prompts. The NPIMOD input prompts are exhibited in Figure D-5.1. The first prompt lets the user enter an initial estimate for the proportionality constant. The user should choose a number between 0 and 1. The final prompt allows the user to enter the maximum number of iterations.

D-5.4. Model Outputs. If a solution is found before the maximum number of iterations is reached, successful convergence output occurs (Figure D-5.2). Otherwise, the user is allowed to repeat execution of the model.

```
PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY
  CONSTANT  (A NUMBER BETWEEN ZERO AND ONE).

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-5.1. NPIMOD Input Prompts.

```
MODEL ESTIMATES AFTER ___ ITERATIONS ARE:
  PROPORTIONALITY CONSTANT OF THE MODEL IS  ________________
  THE TOTAL NUMBER OF ERRORS IS             ________________
PLEASE ENTER 1 FOR AN ESTIMATE OF THE RELIABILITY OF
  THE PROGRAM FOR A SPECIFIED OPERATIONAL TIME BASED
  ON THE CURRENT TESTING EFFORT; ELSE ZERO.
  PLEASE ENTER THE SPECIFIED OPERATIONAL TIME.
  THE ESTIMATED PROBABILITY THAT THE PROGRAM WILL
    OPERATE WITHOUT ERROR FOR THE INPUT TIME IS ________________
  PLEASE ENTER 1 TO TRY ANOTHER OPERATIONAL TIME; ELSE ZERO.
PLEASE ENTER 1 FOR AN ESTIMATE OF THE TESTING TIME REQUIRED
  TO ACHIEVE A SPECIFIED RELIABILITY FOR A SPECIFIED OPERATIONAL
  TIME; ELSE ZERO.
  ENTER DESIRED RELIABILITY AND SPECIFIED OPERATIONAL TIME.
  THE REQUIRED TESTING TIME TO ACHIEVE THE DESIRED RELIABILITY
    FOR THE SPECIFIED OPERATIONAL TIME IS ________________
  PLEASE ENTER 1 TO TRY DIFFERENT VALUES; ELSE ZERO.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-5.2. NPIMOD Successful Convergence Output.

D-6. THE GENERALIZED POISSON MODEL.

D-6.1. Model Description. This model, which is one of the four models within SMERFS that obtains reliability estimates and predictions for interval data, is analogous in form to other models such as the Jelinski-Moranda, Lipow, and Schick-Wolverton models. It is documented in a report by Schafer, Alter, Angus, and Emoto written under contract to the Rome Air Development Center (RADC).

D-6.2. Model Assumptions. The model makes the following five assumptions:

a. The software is operated in a way similar to its expected operational usage.

b. In any time interval, the expected number of discovered software errors is proportional to the product of the total number of existing software errors and to some function of the amount of time spent in testing for software errors. The function is expressed as an exponential function; however, the function could be a linear or parabolic function to allow for a broader class of adaptability.

c. All errors occur with the same probability, and the chance of any given error occurring in no way affects the occurrence or lack of occurrence of any other error (i.e., the errors are independent of each other).

d. The severity of each error is equal.

e. At the end of the testing intervals, errors are corrected without introducing new errors.

D-6.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS GPOMOD module.

D-6.3.1. DATINP Inputs. The data input through the SMERFS module consists of the lengths of the various testing intervals and the number of software faults discovered in each testing interval.

D-6.3.2. GPOMOD Prompts. GPOMOD prompts consist of description and list, correction vector creation, input, and prediction vector creation prompts.

D-6.3.2.1. GPOMOD Description and List Prompt. GPOMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-6.3.2.2. GPOMOD Correction Vector Creation Prompts. SMERFS prompts for a flag which indicates whether or not software fault corrections were performed in the same interval in which they were detected. An error correction vector is created if all error detections and corrections happened during the same intervals; otherwise, the user must enter the number corrected at the end of each period of testing.

D-6.3.2.3. GPOMOD Input and Prediction Vector Creation Prompts. The GPOMOD input prompts are exhibited in Figure D-6.1. The first prompt lets the user terminate model execution or specify the method of estimating (i.e., maximum likelihood or least squares) the model's proportionality constant and the initial total number of errors in the software. After the user enters the desired method, GPOMOD prompts the user for the weighting function or a list of the available functions. If the user desires a list, two weighting functions are listed if least squares was chosen as the method of model parameter estimation; otherwise, one weighting function is listed. The third prompt lets the user specify the weighting function which is either a simple parabolic function or some other polynomial function of order α. If the latter choice is made, GPOMOD will additionally prompt for the order, α, of the polynomial. If the maximum likelihood method was selected earlier, the GPOMOD will prompt the user for an initial estimate of α. In either case, the user is prompted for an initial estimate of the total number of software errors and finally for the maximum number of iterations to be used for the model parameter estimation method.

The GPOMOD prediction vector creation prompts occur later upon successful convergence of the model parameter estimation method.

D-6.4. Model Outputs. If the maximum number of iterations is reached before a solution is found, SMERFS outputs attempted estimates of the model's parameters and the number of remaining software errors. If processing errors occur, then appropriate error messages are output. In either case, the user is allowed to try again. If the model successfully converges to a solution, the output seen in Figure D-6.2 occurs.

```
    PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST
      SQUARES, OR 3 TO TERMINATE MODEL EXECUTION.

    PLEASE ENTER THE WEIGHTING FUNCTION NUMBER, OR ZERO
      FOR A LIST.

      THE AVAILABLE WEIGHTING FUNCTIONS ARE
        1 X(I) ** 2 / 2 (SCHICK-WOLVERTON MODEL)
        2 X(I) ** ALPHA (WHERE ALPHA IS INPUT)

        IF THE MAXIMUM LIKELIHOOD METHOD WAS SELECTED, THEN:

        3 X(I) ** ALPHA (WHERE ALPHA IS ESTIMATED)

        END IF

    PLEASE ENTER THE WEIGHTING FUNCTION NUMBER.

IF AN ALPHA INPUT FUNCTION WAS SELECTED, THEN:

     PLEASE ENTER THE DESIRED ALPHA.

ELSE, IF THE ALPHA ESTIMATION FUNCTION WAS SELECTED, THEN:

     PLEASE ENTER AN INITIAL ESTIMATE FOR ALPHA.

END IF

    PLEASE ENTER AN INITIAL ESTIMATE OF THE TOTAL NUMBER OF
      ERRORS.

    PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-6.1. GPOMOD Input Prompts.

```
IF THE MAXIMUM LIKELIHOOD METHOD (OTHER THAN ALPHA ESTIMATED) WAS
SELECTED, THEN:
  ML MODEL ESTIMATES,  USING THE WEIGHTING FUNCTION TYPE  _, AFTER
   ___ITERATIONS ARE:
   PROPORTIONALITY CONSTANT OF THE MODEL IS ______________
     WITH APP. 95% C.I. OF (______________, ______________)
   THE TOTAL NUMBER OF ERRORS IS            ______________
     WITH APP. 95% C.I. OF (______________, ______________)
   THE REMAINING NUMBER OF ERRORS IS        ______________
     WITH APP. 95% C.I. OF (______________, ______________)
ELSE, IF THE LEAST SQUARES METHOD WAS SELECTED, THEN:
  LS MODEL ESTIMATES,  USING THE WEIGHTING FUNCTION TYPE  _, AFTER
   ___ITERATIONS ARE:
   PROPORTIONALITY CONSTANT OF THE MODEL IS ______________
   THE TOTAL NUMBER OF ERRORS IS            ______________
   THE REMAINING NUMBER OF ERRORS IS        ______________
ELSE, IF THE MAXIMUM LIKELIHOOD METHOD (WITH ALPHA ESTIMATED) WAS
SELECTED, THEN:
  ML MODEL ESTIMATES,  USING THE WEIGHTING FUNCTION TYPE  3, AFTER
   ___ITERATIONS ARE:
   PROPORTIONALITY CONSTANT OF THE MODEL IS ______________
   THE TOTAL NUMBER OF ERRORS IS            ______________
   THE REMAINING NUMBER OF ERRORS IS        ______________
   AND ALPHA IS                             ______________
END IF
  PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS
   EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.
  ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD.
   THE EXPECTED NUMBER OF ERRORS IS ______________
     WITH APP. 95% C.I. OF (______________, ______________)
  PLEASE ENTER 1 TO TRY ANOTHER TESTING LENGTH; ELSE ZERO.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-6.2. GPOMOD Successful Convergence Output.

D-7. GOEL'S NHPP MODEL.

D-7.1. Model Description. This model is one of the four models within SMERFS that obtains reliability estimates and predictions for interval data. It was developed by Amrit Goel and Kazu Okumoto. Following other models, it assumes that counts of software failures over time intervals that don't overlap follow a Poisson distribution. A difference between this model and other Poisson models is that this model treats a program's initial error content as a random variable, and not as a fixed constant.

D-7.2. Model Assumptions. This model makes the following assumptions:

a. The software is operated in a way similar to its expected operational usage.

b. The number of software errors detected in successive time intervals are independent.

c. The probability of detecting any given error is the same as the probability of detecting any other given error. In addition, the severity of each error is assumed to be equal.

d. At any time t, the cumulative number of errors detected follows a Poisson distribution with mean m(t). m(t) satisfies a first order non-homogeneous linear differential equation.

e. m(t) is a bounded, nondecreasing function of t which approaches the expected total number of errors to be detected as t tends to ∞.

D-7.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS NPIMOD module.

D-7.3.1. DATINP Inputs. The data input through the SMERFS DATINP module consists of the lengths of the various testing intervals and the number of software errors discovered in each testing interval.

D-7.3.2. NPIMOD Prompts. NPIMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-7.3.2.1. NPIMOD Description and List Prompt. NPIMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-7.3.2.2. NPIMOD Input and Prediction Vector Creation Prompts. The NPIMOD input prompts are exhibited in Figure D-7.1. The first prompt lets the user terminate model execution or specify the method of estimating (i.e., maximum likelihood or least squares) the model's proportionality constant and the total number of errors in the software. The second prompt allows the user to enter an initial estimate for the model's proportionality constant. A number between 0 and 1 must be chosen to guarantee convergence of the solution. It is recommended that the user choose a small number first, say 0.05 or 0.1, and

```
PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST SQUARES,
  OR 3 TO TERMINATE MODEL EXECUTION.

PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY
  CONSTANT (A NUMBER BETWEEN ZERO AND ONE).

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-7.1. NPIMOD Input Prompts.

then gradually increase it. The last prompt in the first menu lets the user enter the maximum number of iterations to use for the estimation method selected.

The NPIMOD prediction vector creation prompts occur later upon successful convergence output of the model. Through these prompts, NPIMOD lets the user compute predicted interval error counts.

D-7.4. Model Outputs. If the maximum number of iterations is reached before a solution is found, maximum iteration output occurs unless a processing error happens. If the model successfully converges to a solution, the output seen in Figure D-7.2 occurs. If maximum likelihood is chosen, then ML estimates are shown; otherwise, least squares estimates are output. In either event, the user is allowed to estimate the number of expected errors in the next testing period. Figure D-7.2 shows ensuing output if the user does want an estimate of the number of expected errors in the next testing period.

```
IF THE MAXIMUM LIKELIHOOD METHOD WAS SELECTED, THEN:

  ML MODEL ESTIMATES AFTER __ ITERATIONS ARE:
    PROPORTIONALITY CONSTANT OF THE MODEL IS ____________
      WITH APP. 95% C.I. OF (____________,   ____________
    THE TOTAL NUMBER OF ERRORS IS             ____________
      WITH APP. 95% C.I. OF (____________,   ____________

ELSE, IF THE LEAST SQUARES METHOD WAS SELECTED, THEN:

  LS MODEL ESTIMATES AFTER __ ITERATIONS ARE:
    PROPORTIONALITY CONSTANT OF THE MODEL IS ____________
    THE TOTAL NUMBER OF ERRORS IS             ____________

END IF

  PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS
    EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.

    ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD.

    THE EXPECTED NUMBER OF ERRORS IS ____________

    PLEASE ENTER 1 TO TRY ANOTHER TESTING LENGTH; ELSE ZERO.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-7.2. NPIMOD Successful Convergence Output.

D-8. BROOKS AND MOTLEY'S MODEL.

D-8.1. Model Description. This model actually consists of four models each of which obtains reliability estimates and predictions for interval data. They were developed by Brooks and Motley of IBM and include the following: Binomial and Poisson Models for a component of a program and Binomial and Poisson Models for a program. Each of these models accounts for unequal testing of programs in a given testing period.

D-8.2. Model Assumptions. Each model makes the following assumptions:

a. The software is operated in a way similar to its expected operational usage.

b. The ratio of the number of errors reintroduced during the software correction process to the number of errors that are detected is constant.

c. The probability of detecting any error during a given unit interval of testing is constant for any occasion and independent of error detections. The constant is denoted as q in the case of the binomial model, and Φ for the Poisson model.

D-8.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS BAMMOD module.

D-8.3.1. DATINP Inputs. The data input through the SMERFS DATINP module consists of the lengths of the various testing intervals and the number of software errors discovered in each testing interval.

D-8.3.2. BAMMOD Prompts. BAMMOD prompts consist of description and list, fraction of code under test, extended description and list, input, and prediction vector creation prompts.

D-8.3.2.1. BAMMOD Description and List Prompts and Fraction of Code Under Test Prompt. BAMMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP. In addition, BAMMOD has extended description and list prompts which provide extended descriptions of the Binomial and Poisson Models. The fraction of code under test prompt lets the user compensate for partial software testing.

D-8.3.2.2. BAMMOD Input and Prediction Vector Creation Prompts. The BAMMOD input prompts are exhibited in Figure D-8.1. The first prompt lets the user select the appropriate model of interest (Binomial or Poisson) or terminate model execution. The second prompt allows the user to either input or select an initial estimate for α, the probability of correcting errors without inserting new ones. If a decision is made to input α, a suggested range is 0.85-0.95 if no prior knowledge is available. In either event, the total number of errors and the error detection probability are then estimated. The behavior of the estimation process can be observed by trying both low values, such as 0.05-0.1, and high values, such as 0.85-0.90 for the error detection

```
PLEASE ENTER 1 FOR THE BINOMIAL MODEL, 2 FOR THE POISSON
  MODEL, OR 3 TO TERMINATE MODEL EXECUTION.

PLEASE ENTER 1 TO INPUT ALPHA (THE PROBABILITY OF
  CORRECTING ERRORS IN THE PROGRAM WITHOUT INSERTING
  NEW ERRORS), OR 2 TO ESTIMATE ALPHA.

IF ALPHA IS TO BE INPUT, THEN:

  PLEASE ENTER THE DESIRED ALPHA.

  PLEASE ENTER INITIAL ESTIMATES FOR THE TOTAL NUMBER
    OF ERRORS AND THE ERROR DETECTION PROBABILITY.

ELSE, IF ALPHA IS TO BE ESTIMATED, THEN:

  PLEASE ENTER INITIAL ESTIMATES FOR THE TOTAL NUMBER
    OF ERRORS, THE ERROR DETECTION PROBABILITY, AND
    ALPHA.

END IF

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-8.1. BAMMOD Input Prompts.

probability. The last prompt lets the user enter the maximum number of convergence iterations to use for the maximum likelihood estimation method of computing the model parameters.

The BAMMOD prediction vector creation prompts occur later upon successful convergence output of the model. Through these prompts, BAMMOD lets the user compute predicted interval error counts.

D-8.4. Model Outputs. If the maximum number of iterations is reached before a solution is found, maximum iteration output occurs unless a processing error happens. If the model successfully converges to a solution, the output seen in Figure D-8.2 occurs. BAMMOD solutions are based upon maximum likelihood estimates. The last estimate in the figure will be listed only if the user selected alpha estimation. Observing the lower portion of the figure, one may see that SMERFS allows for the optional prediction of errors in the next testing period.

```
THE ______ MODEL WITH ______ ESTIMATES, AFTER ___ INTERATIONS
  ARE:
  PROBABILITY OF DETECTING ERRORS                    ______________
  THE TOTAL NUMBER OF ERRORS IS                      ______________

  IF ALPHA WAS ESTIMATED, THEN:

  PROB.  OF CORRECTING ERRORS WITHOUT ERROR ______________

  END IF

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS
  EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.

  ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD.

  ENTER THE FRACTION OF THE PROGRAM TO BE TESTED
   (FOR FULL PROGRAM, ENTER A 1).

  HOW MANY ERRORS HAVE BEEN FOUND TO DATE IN THE SECTION
   OF THE CODE TO BE TESTED.

  THE EXPECTED NUMBER OF ERRORS IS ______________

  PLEASE ENTER 1 TO TRY ANOTHER TESTING LENGTH; ELSE ZERO.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-8.2. BAMMOD Successful Convergence Output.

D-9. NORMAN SCHNEIDEWIND'S MODEL.

D-9.1. Model Description. This model is another one of the four models within SMERFS that obtains reliability estimates and predictions for interval data. It was developed by Norman Schneidewind. The model theorizes that recent error counts are generally more useful than earlier ones when predicting future error counts because the error detection process changes as testing progresses over time. The model employs three approaches in utilizing error count data:

a. Use all error counts for all m intervals of testing.

b. Completely ignore error counts from the first s - 1 intervals of testing where $2 \leq s \leq m$. Only data from intervals s through m are considered.

c. For intervals 1 through s - 1 use the cumulative error count. For interval s through m, use the individual error counts.

D-9.2 Model Assumptions. This model makes the following assumptions:

a. The software is operated in a way similar to the way it is expected to be used.

b. All errors are independent and occur with equal probability.

c. The ratio of the error correction rate to the number of errors to be corrected is constant.

d. As testing progresses, the mean number of errors that are detected decreases from one interval to the next.

e. The length of each testing period is of the same duration.

f. At the time of the test, the ratio of the rate of error detection to the number of errors within the program is constant. The process of error detection follows a non-homogeneous Poisson process where the error detection rate decreases exponentially.

D-9.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS SDWMOD module.

D-9.3.1. DATINP Inputs. The data input through the SMERFS DATINP consists of the number of software errors discovered in each testing interval.

D-9.3.2. SDWMOD Prompts. SDWMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-9.3.2.1. SDWMOD Description and List Prompts. SDWMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP. The description prompt also includes a description of the three approaches for utilizing the error count data. SMERFS refers to these three approaches as the three treatment types.

D-9.3.2.2. SDWMOD Input and Prediction Vector Creation Prompts. The SDWMOD input prompts are exhibited in Figure D-9.1. The first prompt lets the user terminate model execution or specify one of the three treatment types. If treatment type is 2 or 3, then the user must also enter the associated value of s. The user is then prompted for an initial estimate of the β parameter in the formula for the mean number of errors for the i-th period of testing. Finally, the user is prompted for the maximum number of iterations to use for the maximum likelihood method.

```
PLEASE ENTER THE DESIRED MODEL TREATMENT NUMBER, OR A 4
  TO TERMINATE MODEL EXECUTION.

IF THE TREATMENT TYPE IS 2 OR 3, THEN:

  PLEASE ENTER THE ASSOCIATED VALUE OF S.

END IF

PLEASE ENTER AN INITIAL ESTIMATE FOR THE PARAMETER BETA,
  WHERE THE MEAN NUMBER OF ERRORS FOR THE I-TH PERIOD
  IS TAKEN AS:
  MEAN(I) = ALPHA*(EXP(-BETA(I-1)) - EXP(-BETA(I)))/BETA.

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-9.1. SDWMOD Input Prompts.

The SDWMOD prediction vector creation prompts occur later upon successful convergence output of the model. Through these prompts, SDWMOD lets the user compute predicted interval error counts.

D-9.4. Model Outputs. If the maximum number of iterations is reached before a solution is found, maximum iteration output occurs unless a processing error happens. If the model successfully converges to a solution, the output seen in Figure D-9.2 occurs. The α and β parameters of the error detection rate formula and the weighted sum of squares between the predicted and observed error counts are output. This latter quantity helps decide which treatment type is best. Output also includes an estimate of the number of errors expected in the next testing period and the number of testing periods needed to discover the next M errors, where M is specified by the user.

```
TREATMENT __ MODEL ESTIMATES AFTER ___ ITERATIONS ARE:
  BETA ___________
  ALPHA ___________
AND THE WEIGHTED SUMS - OF - SQUARES BETWEEN THE PREDICTED
  AND OBSERVED ERROR COUNTS IS ___________

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS
  EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.

  THE EXPECTED NUMBER OF ERRORS IS ___________

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF
  TESTING PERIODS NEEDED TO DISCOVER THE NEXT
  M ERRORS; ELSE ZERO.

  PLEASE ENTER THE VALUE FOR M.

  THE EXPECTED NUMBER OF PERIODS IS ___________

  PLEASE ENTER 1 TO TRY A DIFFERENT VALUE FOR M;
  ELSE ZERO.

*** THE ESTIMATE CANNOT BE MADE FOR THE SPECIFIED M VALUE.
```

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-9.2. SDWMOD Successful Convergence Output.

APPENDIX E

GLOSSARY

E-1. SCOPE

The following terms are identified and defined as they are used throughout the Software Maturity Model Investigation.

Binomial distribution

A discrete probability function whose terms correspond to successive terms in the binomial expansion.

Chi-square distribution

A special case of the gamma distribution.

Coefficient of Kurtosis

A measure of the perkiness of a distribution. A large coefficient of kurtosis for a distribution indicates that the values of the distribution are concentrated near the mean.

Coefficient of Skewness

A measure of the asymmetry of a distribution. A distribution whose longer tail occurs to the left is said to be skewed to the left, whereas a distribution whose longer tail occurs to the right is said to be skewed to the right.

Confidence interval

Specifies a statistical range of values for some parameter. The parameter being estimated is said to lie within confidence limits which express the degree of confidence.

Cumulative distribution function

For a random variable X it is the probability that $X \leq x$ where x is any real number, such that $-\infty < x < \infty$.

Distribution

A probability function or a cumulative distribution function.

exp

An abbreviation for a function expressed in terms of powers of e, the base of natural logarithms.

Exponential distribution

The distribution of a random variable whose corresponding probability density function is a certain exponential function.

Exponential function

A mathematical function expressed in terms of a power of the base of natural logarithms, e.

Gamma Distribution

A random variable has the gamma distribution if the probability density function is a certain complex mathematical function involving the gamma function.

Gamma Function

A complex mathematical function defined in terms of what is known in calculus as an integral function.

Geometric Progression

A geometric progression is a sequence of numbers where the ratio of any given number to the one that precedes is equal to a constant known as the common ratio.

Hazard Rate

The conditional probability that a software failure occurs in an interval of time given that the software has not failed up to the beginning of that time interval.

Least Squares Estimation

A method of approximating or fitting data by some mathematical function. The approximating function is known as the least-squares approximation.

Limit

A function $f(x)$ of a single variable x approaches a number L as a limit if for any positive number ϵ there exists a positive number δ such that the absolute value of $f(x) - L < \epsilon$ whenever $0 < |x - x_0| < \delta$.

Linear function

An algebraic function in which the highest degree term in the variable(s) is of the first degree.

Maximum Likelihood Estimation

A method for estimating parameters of a mathematical function. The mathematical function is known as the likelihood function. Techniques of calculus are used to maximize this function for one or more parameters being determined. A system of simultaneous equations is then solved to determine the parameters.

Mean

The mathematical expectation for a discrete or random variable is referred to as the mean of that random variable. Often represented by the Greek letter μ.

Mean Time Between Failure

The expected time between one error occurrence and another.

Mean Time To Repair

The expected time between the occurrence of an error and its repair.

Nonhomogeneous Linear Differential Equation

A linear differential equation whose right side is a nonzero function of the independent variable.

Non-Homogeneous Poisson Process

A random process (e.g., of software failures) whose value (e.g., time of occurrence of software failure) at each point in time follows a Poisson probability distribution that itself varies with time.

Parameter

A variable or constant which appears in a mathematical expression. The specific form of the expression is determined by the value of the constant or variable.

Poisson Distribution

A discrete probability distribution whose probability function is a certain mathematical function discovered by S. D. Poisson in the 19th century.

Probability

A measure of the chance that an event will or will not occur. This measure will always be a number between 0 and 1.

Probability Density Function

A continuous probability function for one or more continuous random variables.

Probability Distribution

A probability function.

Probability Function

A probability function of a single random variable is either discrete or continuous. In either case it is a nonnegative number. In the discrete case, the sum of all possible functional values equals 1. In the continuous case, the improper integral of the function is 1. These ideas are generalized to two or more random variables giving us joint probability distributions for the discrete and continuous cases.

Quadratic function

An algebraic function possessing quantities of the second degree or less.

Random Variable

A variable, in the mathematical sense, which takes on numerical values associated with the outcome of a chance experiment.

Remaining Number of Errors

The number of software errors remaining in a program.

Software Error

A human error which introduces a fault in software.

Software Failure

A deviation of the operation of software from its requirements. It is caused by a software fault.

Software Fault

A defect in software which causes a software failure to occur when that software is executed.

Software Maturity

This is defined in AFOTECP 800-2 Volume 1, 1 August 1986, as "a measure of the software's evolution toward satisfying all documented user requirements."

Software Performance Parameter

An objectively quantifiable measure of an aspect of software behavior.

Software Quality Indicator

According to AMC-P 70-14, 30 April 1987, software quality indicators are "quality indicators designed for and specifically allied to software projects." AMC-P 70-14 further defines quality indicators as "process guidelines in the form of detailed data, derived from scheduled surveys, inspections, evaluations, and tests, that provide insight into the condition of a product or process."

Software Reliability

This is defined in William Farr's survey of software reliability modeling and estimation as "the probability that a given software program will operate without failure for a specified time in a specified environment."

Standard Deviation

This is the positive square root of the variance of a random variable. Often denoted by the Greek letter σ.

Time Between Error Occurrences

This simply refers to the difference between the points in time at which errors occur.

Variance

The variance is a measure of the dispersion that values of a random variable have about their mean. A small variance indicates a concentration of values near the mean, whereas a large variance indicates a tendency for values to be scattered far from the mean. The variance is a number that is nonnegative.

This Page Intentionally Blank

APPENDIX F

DISTRIBUTION

Addressee	Number of Copies
Director	
U.S. Army Materiel Systems Analysis Activity	
ATTN: AMXSY-MP	1
Aberdeen Proving Ground, MD 21005-5071	
Commander	
U.S. Army Test and Evaluation Command	
ATTN: AMSTE-EV-S	1
ATTN: AMSTE-TC-M	3
ATTN: AMSTE-TE	6
ATTN: AMSTE-TO	2
Aberdeen Proving Ground, MD 21005-5055	
Commander	
Defense Technical Information Center	
ATTN: FDAC	2
Cameron Station	
Alexandria, VA 22304-6145	
Commander	
U.S. Army Cold Regions Test Center	
ATTN: STECR-TM	1
APO Seattle, WA 98733-5000	
Commander	
U.S. Army Combat Systems Test Activity	
ATTN: STECS-DA-M	2
Aberdeen Proving Ground, MD 21005-5000	
Commander	
U.S. Army Dugway Proving Ground	
ATTN: STEDP-PO-P	1
Dugway, UT 84022-5000	
Commander	
U.S. Army Electronic Proving Ground	
ATTN: STEEP-TD	1
ATTN: STEEP-ET	1
ATTN: STEEP-DT	1
ATTN: STEEP-MO	4
Fort Huachuca, AZ 85613-7110	

Addressee	Number of Copies
Commander U.S. Army Jefferson Proving Ground ATTN: STEJP-TD-E Madison, IN 47250-5000	1
Commander U.S. Army Tropic Test Center ATTN: STETC-TD-AB APO Miami, FL 34004-5000	1
Commander U.S. Army White Sands Missile Range	
ATTN: STEWS-TE-A	1
ATTN: STEWS-TE-M	1
ATTN: STEWS-TE-O	1
ATTN: STEWS-TE-PY	4
White Sands Missile Range, NM 88002-5000	
Commander U.S. Army Yuma Proving Ground ATTN: STEYP-MSA Yuma, AZ 85634-5000	2